# Graph Filtration Surfaces

## A ~~Deep~~ Multi-Scale Approach to Dynamic Graph Representation

*by*

Franz Srambical

Bachelor's Thesis in Informatics

*School of Computation, Information and Technology - Informatics*

at

Technical University of Munich

# Graph Filtration Surfaces

# Filtrationsflächen zur Graphenrepräsentation


| | |
|---|---|
| Author: | Franz Srambical |
| Supervisor: | Prof. Dr. Niki Kilbertus |
| Advisor: | Dr. Bastian Rieck |
| Submission Date: | August 15, 2023 |



Bachelor's Thesis in Informatics
*School of Computation, Information and Technology - Informatics*
at
Technical University of Munich

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, August 10, 2023 Franz Srambical

## Abstract

Most of the existing approaches for classifying dynamic graphs either lift graph kernels to the temporal domain, or use graph neural networks. While the former approach achieves state-of-the-art accuracies, its scalability is severely limited due to the blow-up during the "lifting" operation and the computational complexity of graph kernels. The latter approach does not perform as well on dynamic-graph-level classification tasks, since it is designed for link prediction and node classification. In this work, we propose a novel method for dynamic graph classification, which we term *filtration surfaces*. We experimentally validate the efficacy of our model and show that filtration surfaces outperform methods that act on dynamic graphs that are converted to spatiotemporal variants, and attain comparable accuracies to graph neural networks applied to static graphs obtained by a sophisticated conversion scheme that minimizes the loss of temporal information. Our method does so while being either completely parameter-free or having at most one parameter, and yielding the lowest overall standard deviation. We also describe how filtration surfaces can be adapted to better model a setting where the input dynamic graph is obtained by sampling from an underlying "ground truth" dynamic graph and propose an approach for extrapolating filtration surfaces to unseen, future time points. Lastly, we introduce a second, related, but highly scalable method which we term *filtration trajectories* and prove its expressivity upper-bound.

# Contents


1 Introduction 1

2 Background 3
2.1 Static Graphs . . . 3
2.2 Dynamic Graphs . . . 3
2.2.1 Taxonomy based on Edge Lifetimes . . . 3
2.2.2 Modelling Dynamic Graphs . . . 3
2.3 Filtrations . . . 4

3 Related Work 5
3.1 Dynamic Graph Neural Networks . . . 5
3.2 Graph-Shapelet Pattern . . . 5
3.3 Stable Distance Persistent Homology . . . 5
3.4 Temporal Graph Kernels . . . 6

4 Methods 7
4.1 Filtration Curves . . . 7
4.1.1 Edge Weight Function . . . 7
4.1.2 Graph Descriptor Function . . . 9
4.1.3 Properties of Filtration Curves . . . 9
4.2 Filtration Surfaces . . . 10
4.2.1 Classification . . . 10
4.2.2 Filtration Trajectories . . . 10

5 Experiments 15
5.1 Experimental Setup . . . 15
5.2 Synthetic Datasets . . . 15
5.3 Real-World Datasets . . . 15

6 Conclusion 21

Bibliography 23

Appendices 27

Additional Figures 29

# 1 Introduction

Recent years have seen a wealth of research on the analysis of graph-structured data [41] [45] [50] [36]. Graphs are a natural way to represent permutation-invariant data, such as social networks [29], citation networks [21], or protein-protein interaction networks [11]. While the subfield of static graph analysis has been studied for decades, the analysis of *dynamic* graphs is still a relatively nascent field. In this work, we will focus on the task of dynamic graph classification, where the goal is to predict the class of a process, which cannot be observed from a single snapshot of the graph, but only from the evolutionary pattern of the graph over time. Examples for such processes include the spread of a disease in an interaction network, the evolution of a citation network or the growth of different types of social communities.

A central challenge in working with dynamic graph data involves learning appropriate representations of a graph's evolution over time. Over the years, numerous strategies have been proposed, including Neural Temporal Walks [12], RNNs [35] and Temporal Point Processes [39]. In this work, we will explore a different approach based on *filtrations*, a concept from topological data analysis typically associated with persistent homology [8]. Specifically, we will be extending prior work from O'Bray, Rieck, and Borgwardt on *filtration curves* [23] to a dynamic setting.

The remainder of this thesis is structured as follows: We first provide some background on graphs and clarify the terminology used in dynamic graph learning literature. We then review related work on dynamic graph classification and existing approaches. Subsequently, we introduce our proposed method, which we term *filtration surfaces*. We will experimentally validate the efficacy of our method and conclude with a discussion of our results and potential future work.

Our code and instructions to reproduce our experiments are available at `https://github.com/emergenz/filtration_surfaces`.

# 2 Background

## 2.1 Static Graphs

A (static) undirected graph $G = (V, E)$ is a structure consisting of $N$ pairwise different nodes $V = \{v_1, \ldots, v_N\}$ and edges $E \subseteq \{X \in \mathcal{P}(V) : |X| = 2\}$. Graphs are called *directed* when edges have a directionality associated with them, and *undirected* otherwise. A *labeled* graph additionally possesses a label function $l_V : V \to \Sigma$, where $\Sigma$ is some alphabet. Edges can also be labeled via a label function $l_E : E \to \Gamma$. When $\Sigma = \mathbb{R}^d$ for some $d \in \mathbb{N}$, the label of a node $l_V(v)$ is called the (continuous) attribute of $v$. When $\Gamma = \mathbb{R}$, $l_E(e)$ is called the weight of the edge $e$. A graph is called *heterogeneous* if it has a discrete (node or edge) label function, and *homogeneous* otherwise [15].

## 2.2 Dynamic Graphs

Dynamic graphs can be formulated and represented in many different ways. To bring clarity to the terminology used in the literature, we will first introduce a distinction based on lifetimes of edges, which we adapt from [31].

### 2.2.1 Taxonomy based on Edge Lifetimes

**Interaction graphs:** An interaction graph is a dynamic graph where edges are instantaneous events. This type of graph is often represented as a contact sequence [20].

**Temporal graphs:** Temporal graphs have short-lived (albeit non-instantaneous) edges. Edges can be thought of as having a duration.

**Evolving graphs:** In evolving graphs, edges are generally long-lived. It is more natural to think of the appearance and disappearance of edges in an evolving graph as two separate events.

### 2.2.2 Modelling Dynamic Graphs

We will now introduce multiple modelling approaches for dynamic graphs.

**Spatiotemporal graphs:** In spatiotemporal graphs only the node or edge features change, while the graph topology stays the same. The temporal dimension is encoded in the edge features.

**Discrete-time dynamic graphs:** Discrete-time dynamic graphs are defined as a finite sequence of static graphs $\mathcal{G} = (\mathcal{G}_1, \ldots, \mathcal{G}_n)$ and represent $n$ snapshots of the dynamic graph in discrete time intervals which are assumed to be of equal length.

**Continuous-time dynamic graphs:** These constitute the most general type of dynamic graph representation and are usually modelled as a sequence of time-stamped events $\mathcal{G} = (x(t_1), x(t_2), \ldots)$, where $x$ can be a node or an edge operation [28]. Operations can be inserts, updates or deletions. Another more recent approach is to model a continuous-time dynamic graph as an adjacency matrix containing posets (partially-ordered sets) of edge lifetimes [3]. While similar to the aforementioned method, this permits exploitation of the semi-ring properties of said matrix. We did not explore the latter approach, but deem it a promising avenue for future work.

## 2.3 Filtrations

While filtrations are a general topological method, we restrict ourselves to filtrations on graphs and refer the reader to [8] for a more general introduction.

A filtration $\mathcal{F}_G$ on a graph $G = (V, E)$ is a sequence of subgraphs $(G_1, \ldots, G_m)$ such that $\emptyset \subseteq G_1 \subseteq \ldots \subseteq G_m = G$. A filtration can be obtained by iteratively adding edges to the starting graph $G_1$. More specifically, a filtration is defined using an edge weight function $w : E \to \mathbb{R}$ such that $G_i$ is induced by all edges with weights less than or equal to $w_i$, where $w_i$ is the i$^{\text{th}}$ smallest weight. This means that creating a filtration is equivalent to sorting the edges by weight and progressively adding them to the graph in order of increasing weight, yielding a time complexity of $\mathcal{O}(n \log n)$ [23].

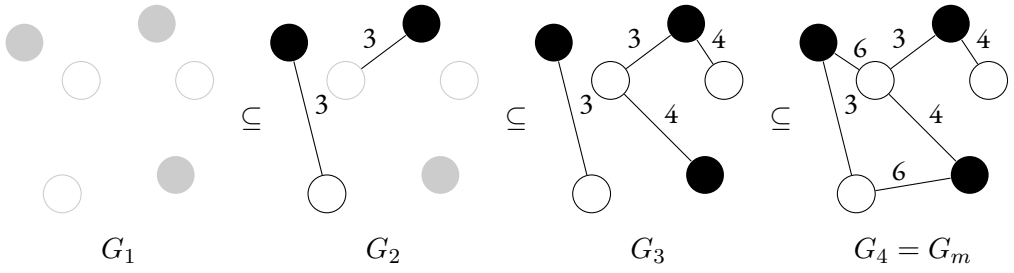



Figure 2.1: Filtration of a graph. Black and white nodes depict nodes with different attributes. Edges are consecutively added to the graph along the filtration in order of increasing weight.

# 3 Related Work

## 3.1 Dynamic Graph Neural Networks

In recent years, there has been a notable surge in the popularity of methods analysing dynamic graphs [51] [38] [10]. Among these methods, a significant portion of research efforts has been dedicated to dynamic graph neural networks (DyGNNs), which are specifically designed to excel in link prediction and node classification tasks [31]. Although DyGNNs can handle dynamic graph classification via global readout functions, their performance on such tasks remains largely unexplored in the literature, with only a few isolated exceptions noted [19] [44] [42]. One such exception, GDGESN [18], works by combining a Dynamical Graph Echo State Network (DynGESN) [37] with snapshot merging, resulting in accuracies that surpass those of the vanilla DynGESN, albeit falling short of the performance achieved by temporal graph kernels. Another exception, STAGIN [14], addresses the issue of the limited expressivity of vanilla readout functions by introducing spatial-attention-based readout functions. Nevertheless, a big drawback of DyGNNs is the need for extensive hyperparameter tuning to attain optimal performance.

## 3.2 Graph-Shapelet Pattern

Wang et al. [43] try to solve the classification problem by first transforming a dynamic graph into a univariate or multivariate time series, and subsequently extracting time-series shapelets [49]. Shapelets are time series subsequences, which are maximally useful for distinguishing between classes. Using shapelets instead of entire time series allows for a significantly more efficient, nicely interpretable and often more robust classification, as noise that stems from subsequences that are disjoint from the shapelets is ignored. Shapelets are extracted using brute-force shapelet search or more efficient techniques such as *Subsequence Distance Early Abandon* or *Early Entropy Pruning* [49].

## 3.3 Stable Distance Persistent Homology

Ye et al. introduced a novel representation for dynamic graphs called Stable Distance Persistent Homology (SDPH) [48]. This representation is subsequently input to a kernel machine, such as a support vector machine, to perform classification. The SDPH representation is obtained by first computing persistence diagrams using a dynamic Dowker filtration of the dynamic graph, and then vectorizing these diagrams using a method the authors coined Time-interlevel Kernel. We will not elaborate on the details of this method and refer the curious reader to the original paper.

## 3.4 Temporal Graph Kernels

Temporal graph kernels [25] present an entirely different approach to dynamic graph classification by lifting standard graph kernels to the temporal domain. While they achieve state-of-the-art accuracies, it is important to note that (i) the transformation to the static graph can lead to a blow-up of the size of the graph and (ii) the computational complexity is lower-bounded by that of standard graph kernels . As a consequence, the use of this method becomes impractical for larger graphs.

# 4 Methods

Before we introduce our proposed method, we will first provide some background on filtration curves. This will equip us with the necessary tools to find a natural extension to the dynamic setting.

## 4.1 Filtration Curves

Filtration curves [23] are expressive, computationally efficient representations of static graphs. Unlike methods based on subgraph matching or neighbourhood comparison, filtration curves take both edge weights as well as the graph topology into account. The fundamental postulate behind filtration curves is that two graphs generated by a similar process have similar substructures emerging at similar filtration timesteps.

To build a filtration curve, we need to choose (i) an edge weight function $w : E \to \mathbb{R}$ that assigns a weight to every edge, and (ii) a graph descriptor function $f : G \to \mathbb{R}^d$ that takes a (sub)graph and returns a value in $\mathbb{R}^d$. By building the graph filtration $\mathcal{F}_G$ in order of increasing edge weight and evaluating the graph descriptor function on every subgraph $G_i$ of the filtration, we obtain a sequence of vectors which we model as a matrix $\mathcal{P}_G \coloneqq \bigoplus_{i=1}^{m} f(G_i) \in \mathbb{R}^{m \times d}$, the structure that O'Bray et al. [23] termed filtration curve. In this definition, $m$ stands for the number of thresholding edge weights in the filtration $\mathcal{F}_G$, and $\bigoplus$ denotes the concatenation operator. The filtration curve $\mathcal{P}_G$ is a compact representation of the graph $G$ that can be used for downstream tasks such as graph classification. Another way of looking at filtration curves is as a type of (topological) feature extraction method. The graph descriptor function $f$ can be thought of as a feature extractor, which, when evaluated alongside a filtration, yields a multi-scale representation of the graph.

It shall be noted that the weight thresholds of $\mathcal{P}_G$ are not necessarily equal among all graphs in the dataset. This is a very deliberate design choice as it allows for a sparse representation of the graph with only changes of the curve at the weight thresholds being stored in the matrix. However, it is also possible to standardize the representations by creating a shared sorted index of all weight thresholds of the dataset and forward-filling the missing values.

In the following sections, we will discuss the choice of edge weight and graph descriptor functions in more detail.

### 4.1.1 Edge Weight Function

The edge weight function is needed in order to define a filtration over the graph. O'Bray et al. propose the following edge weight functions:

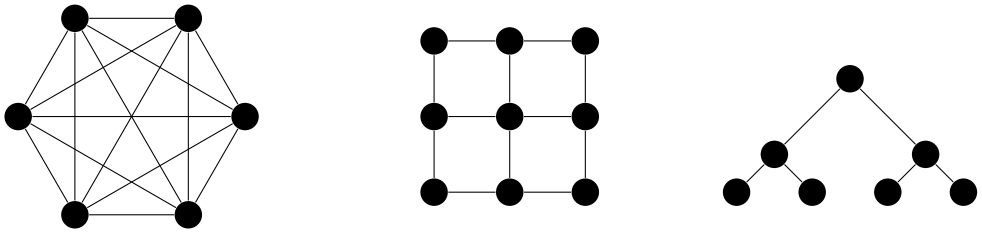


Figure 4.1: Examples of graphs with different Ricci curvature values. The leftmost graph has positive curvature, the middle graph has zero curvature, and the rightmost graph has negative curvature.

### Native Edge Weights

If the graph $G$ has an edge label function $l_E : E \to \mathbb{R}$, we can simply use these labels as edge weights.

### Max Degree

The max degree edge weight function $w_{ij} = \max\{\text{degree}(i), \text{degree}(j)\}$ assigns to every edge the maximum degree of its incident nodes. However, the authors note the limited expressiveness of this function, as no further information about vertex neighbourhoods is taken into account.

### Ricci Curvature

Ricci curvature is a graph isomorphism invariant, which means that it is a property that remains unchanged under graph isomorphisms. Two graphs are said to be isomorphic if there exists a bijective mapping between their vertices such that the adjacency relationships are preserved [27]. Intuitively, Ricci curvature measures the deviation of a graph from a grid graph. To give an example, a tree has negative curvature, while a fully connected graph has positive curvature (figure 4.1).

More specifically, the authors use the curvature definition proposed by Ollivier [26]:

$$\kappa_\alpha(x, y) = 1 - \frac{W(m_x^\alpha, m_y^\alpha)}{d(x, y)} \tag{4.1}$$

for all $x, y \in V$. Here, $W$ denotes the Wasserstein distance [13] between two probability measures. O'Bray et al. use the shortest path distance $d$ and a probability measure which is defined as

$$m_x^\alpha(v) = \begin{cases} \alpha & \text{if } v = x \\ \frac{1-\alpha}{\text{degree}(x)} & \text{if } v \in \mathcal{N}(x) \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\alpha \in [0, 1]$ is a parameter that controls the influence of the node degree on the curvature, and $\mathcal{N}(x)$ denotes the set of neighbours of $x$. The authors set $\alpha = 0.5$ and assign the Ollivier-Ricci curvature $\kappa_\alpha(x, y)$ as the weight of each edge $\{x, y\}$.

#### Heat Kernel Signature

The heat kernel signature [34] is a highly expressive, but computationally costly summary due to needing a full eigendecomposition of a matrix. Carrière et al. [4] define it as

$$\mathrm{hks}(G, t, v) = \sum_{i=1}^{n} \exp(-t\lambda_i)\psi_i(v)^2, \tag{4.3}$$

where $t$ is the diffusion parameter, $\lambda_i$ is the $i^{\text{th}}$ eigenvalue of the Laplacian matrix of $G$, and $\psi_i(v)$ is the $i^{\text{th}}$ eigenfunction of the graph Laplacian. The weight of each edge $\{x, y\}$ is then calculated as $\max\{\mathrm{hks}(G, t, x), \mathrm{hks}(G, t, y)\}$.

### 4.1.2 Graph Descriptor Function

Now that we have defined a filtration over the graph, we need to choose a graph descriptor function $f$. O'Bray et al. propose the following graph descriptor functions:

#### Node Label Histogram

The node label histogram graph descriptor function counts the number of nodes with a given label. The dimensionality of the descriptor is equal to the number of unique node labels in the dataset.

#### Count of Connected Components

As the node label histogram cannot be used for graphs without node labels, the authors resort to counting the number of connected components in such cases. This number only changes at thresholds at which a connected component is either created or destroyed. Therefore, it suffices to only store the count at these thresholds, leading to an even sparser representation of the graph.

### 4.1.3 Properties of Filtration Curves

O'Bray et al. note that when decomposing the graph descriptor function $f : G \to \mathbb{R}^d$ into a set of subfunctions $f_i : G \to \mathbb{R}$, one obtains piecewise linear functions that form a vector space. Since this means that addition and scalar multiplication is well-defined for these subfunctions, one can calculate a mean filtration curve. As each of these subfunctions can be decomposed into step functions (which satisfy integrability), the $L^p$-norm is defined as

$$\|f_i\| = \left(\int_{\mathbb{R}} |f_i(x)|^p dx\right)^{\frac{1}{p}}. \tag{4.4}$$

Therefore, a similarity function for two subfunctions $f, g$ is obtained by calculating

$$\langle f, g \rangle = \int_{\mathbb{R}} f(x)g(x)dx. \tag{4.5}$$

Equation 4.5 can be computed efficiently and subsequently input to a kernel method such as an SVM. However, the authors chose to vectorize $\mathcal{P}_G$ instead, as this allows for the use of a random forest classifier while preserving the ability to discover interactions between the subfunctions.

## 4.2 Filtration Surfaces

Now that we have introduced filtration curves, we propose a natural way of extending them to the dynamic setting. Specifically, we propose a method for classifying discrete-time dynamic graphs (DTDG) $\mathcal{G}$. Intuitively, we calculate filtration curves $\mathcal{P}_{\mathcal{G}_i}$ for all graphs $\mathcal{G}_i \in \mathcal{G}$ and therefore extend the curve to a third dimension, yielding a surface. Formally, we model the sequence of filtration curves as a tensor $\mathcal{R}_{\mathcal{G}} := \bigoplus_{i=1}^{n} \mathcal{P}_{\mathcal{G}_i} \in \mathbb{R}^{n \times m \times d}$, where $n$ is the length of the dynamic graph, $m$ is the number of weight thresholds in the filtration, and $d$ is the dimensionality of the graph descriptor function. The careful reader will note that the filtration curves $\mathcal{P}_{\mathcal{G}_i}$ do not necessarily share the same weight thresholds. Therefore, it is necessary to compute a shared weight index as described in section 4.1.

Just like filtration curves are step functions because the graph descriptor function does not change in-between thresholding weights, filtration surfaces can be thought of as step-like surfaces when assuming that the filtration curve does not change in-between timestamps of the dynamic graph. Latter assumption is reasonable when the dynamic graph is the "ground truth" and the direct result of some generation process. However, in the case of real-world data, it is likely that the dynamic graph was obtained by sampling a streaming graph at discrete time intervals. In such cases, forward-filling filtration curves until the next timestamp might not be the optimal approach. Instead, our representation permits the use of any interpolation function to fill the gaps between timestamps, which can even be learnt from data, yielding a (lossy) compression mechanism. Alternatively, one could learn an interpolation function by masking parts of a filtration surface of an underlying "ground truth" dynamic graph and predicting the missing values, similar to the proxy objective of BERT [5], or even mask the entire filtration surface from a specific timestamp onwards and predict the next filtration curve, similar to the objective of the original Transformer architecture [40]. The latter approach would yield a model for filtration surfaces with the ability to extrapolate beyond the observed time interval.

### 4.2.1 Classification

In order to classify filtration surfaces, we vectorize them by flattening the tensor along the time dimension, and flattening the resulting matrix along the weight dimension. The resulting vector is then input to a random forest classifier. This means that each dimension of the input vector corresponds to a specific weight threshold at a specific timestamp. Since we standardized all filtration curves of a dataset to have the same weight thresholds, all vectors have the same length and all vector dimensions are comparable.

### 4.2.2 Filtration Trajectories

Another way of extending filtration curves to the dynamic setting is to model the growth process of a dynamic graph as a filtration along the time dimension to obtain what we term a *filtration*

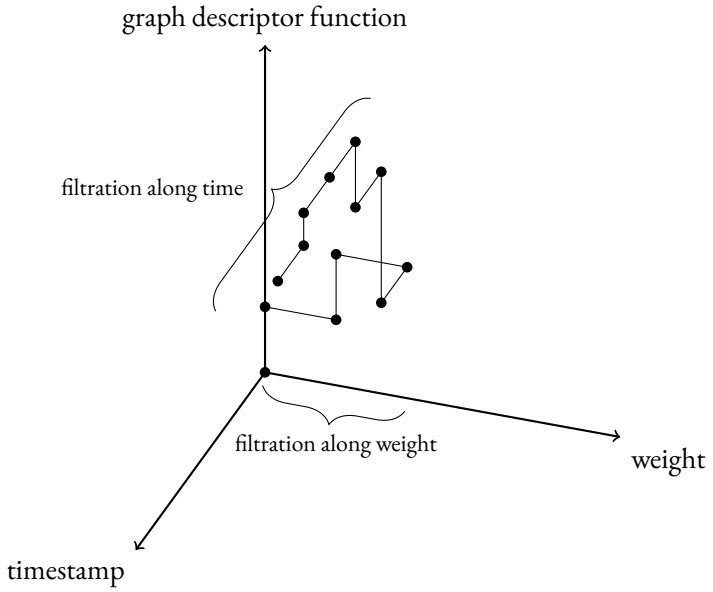



Figure 4.2: Filtration trajectory of a dynamic graph for a 1-dimensional graph descriptor function. If the graph-descriptor function is higher dimensional, we can construct a filtration trajectory for each dimension.

*trajectory*. Intuitively, this means that we extend the $d$-dimensional space of the filtration curve to a $(d+1)$-dimensional space, where the additional dimension is the time dimension. The filtration trajectory then stays in the $d$-dimensional subspace for the duration of the construction of the starting graph of the dynamic graph and subsequently starts moving along the time dimension.

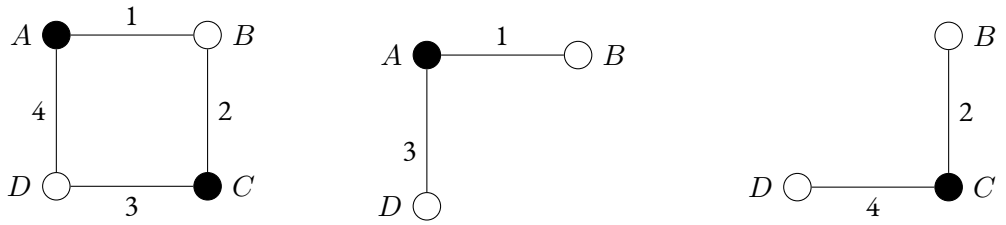
Formally, we model the filtration trajectory as a matrix $\mathcal{T}_{\mathcal{G}} \coloneqq \bigoplus_{i=1}^{m} \hat{f}((\mathcal{G}_1)_i) \oplus \bigoplus_{i=2}^{n} \tilde{f}(\mathcal{G}_i) \in \mathbb{R}^{(m+n-1)\times(d+1)}$, where $\mathcal{G}_1$ is the starting graph (with $(\mathcal{G}_1)_i$ being the $i^{\text{th}}$ subgraph in the filtration of the starting graph), $\mathcal{G}$ is the dynamic graph (with $\mathcal{G}_i$ being the graph at timestamp $i$ of the dynamic graph), $\hat{f} : G \to \mathbb{R}^{(d+1)}$ is the graph descriptor function $f$, but with an additional output dimension which encodes the timestep and is always set to 1, and $\tilde{f} : G \to \mathbb{R}^{(d+1)}$ is the function $\hat{f}$, but with the additional output dimension being the timestamp of the graph $\mathcal{G}_i$.

We will now prove that filtration surfaces are strictly more expressive than filtration trajectories by first proving that filtration surfaces are at least as expressive as filtration trajectories, and subsequently giving an example of dynamic graphs that have distinct filtration surfaces, but indistinguishable filtration trajectories.

**Lemma 1.** *All dynamic graphs that have distinct filtration trajectories also have distinct filtration surfaces.*

*Proof.* Let $\mathcal{G}', \mathcal{G}''$ be two dynamic graphs and let $\mathcal{T}_{\mathcal{G}'}, \mathcal{T}_{\mathcal{G}''}$ be their respective filtration trajectories. Let $\mathcal{G}'_t$, $\mathcal{G}''_t$ be the graphs at timestamp $t$ of $\mathcal{G}'$ and $\mathcal{G}''$ respectively. Then we have two cases: (i) There exists a step during the filtration along the weights of $\mathcal{G}'_1$ and $\mathcal{G}''_1$, and therefore also a weight $w'$, such that $\hat{f}((\mathcal{G}'_1)_{w'}) \neq \hat{f}((\mathcal{G}''_1)_{w'})$. Then it follows per definition that $f((\mathcal{G}'_1)_{w'}) \neq f((\mathcal{G}''_1)_{w'})$: the first filtration curves of the two filtration surfaces are distinct,

and therefore $\mathcal{R}_{\mathcal{G}'}$ and $\mathcal{R}_{\mathcal{G}''}$ are also distinct. (ii) There exists a timestamp $t > 1$ such that $\tilde{f}(\mathcal{G}'_t) \neq \tilde{f}(\mathcal{G}''_t)$. Then there exists a step during the filtration along the weights of $\mathcal{G}'_t$ and $\mathcal{G}''_t$, and therefore also a weight $w''$, such that $f((\mathcal{G}'_t)_{w''}) \neq f((\mathcal{G}''_t)_{w''})$. Specifically, the weight $w''$ is the maximum weight threshold. □




(a) Graph $\mathcal{G}_S$ (b) Graph $\mathcal{G}'_2$ (c) Graph $\mathcal{G}''_2$

Figure 4.3: Graphical representations of $\mathcal{G}_S$, $\mathcal{G}'_2$, and $\mathcal{G}''_2$ of Lemma 2.

**Lemma 2.** *There exist dynamic graphs that have distinct filtration surfaces, but indistinguishable filtration trajectories.*

To prove this, one can construct any two dynamic graphs $\mathcal{G}', \mathcal{G}''$ with arbitrary, but fixed $\mathcal{G}_S = \mathcal{G}'_1 = \mathcal{G}''_1$, and $\mathcal{G}'_2, \mathcal{G}''_2$, such that $\tilde{f}(\mathcal{G}'_2) = \tilde{f}(\mathcal{G}''_2)$ and $\mathcal{P}_{\mathcal{G}'_2} \neq \mathcal{P}_{\mathcal{G}''_2}$. We give one such example.

*Proof.* Let $\mathcal{G}' = (\mathcal{G}_S, \mathcal{G}'_2)$, $\mathcal{G}'' = (\mathcal{G}_S, \mathcal{G}''_2)$ be two dynamic graphs of length 2, where $\mathcal{G}_S = (\{A, B, C, D\}, (\{A, B\}, \{B, C\}, \{C, D\}, \{D, A\}))$.

Let $\mathcal{G}_S$ have a node label function $l_{V,\mathcal{G}_S}(v) = \begin{cases} 1 & \text{if } v \in \{A, C\} \\ 0 & \text{if } v \in \{B, D\} \end{cases}$ and an edge label function $l_{E,\mathcal{G}_S}(e) = \begin{cases} 1 & \text{if } e = \{A, B\} \\ 2 & \text{if } e = \{B, C\} \\ 3 & \text{if } e = \{C, D\} \\ 4 & \text{if } e = \{D, A\} \end{cases}$ (figure 4.3a).

Let $\mathcal{G}'_2 = (\{A, B, D\}, (\{A, B\}, \{D, A\}))$ with $l_{V,\mathcal{G}'_2}(v) = \begin{cases} 1 & \text{if } v = A \\ 0 & \text{if } v \in \{B, D\} \end{cases}$ and $l_{E,\mathcal{G}'_2}(e) = \begin{cases} 1 & \text{if } e = \{A, B\} \\ 3 & \text{if } e = \{D, A\} \end{cases}$ (figure 4.3b).

Let $\mathcal{G}''_2 = (\{B, C, D\}, (\{B, C\}, \{C, D\}))$ with $l_{V,\mathcal{G}''_2}(v) = \begin{cases} 1 & \text{if } v = C \\ 0 & \text{if } v \in \{B, D\} \end{cases}$ and $l_{E,\mathcal{G}''_2}(e) = \begin{cases} 2 & \text{if } e = \{B, C\} \\ 4 & \text{if } e = \{C, D\} \end{cases}$ (figure 4.3c).

Then the (standardized) node label filtration curve of $\mathcal{G}'_2$ is $\mathcal{P}_{\mathcal{G}'_2} = ((1, 1), (1, 1), (2, 1), (2, 1))$, while the (standardized) node label filtration curve of $\mathcal{G}''_2$ is $\mathcal{P}_{\mathcal{G}''_2} = ((0, 0), (1, 1), (1, 1), (2, 1))$.

Since $\mathcal{P}_{\mathcal{G}'_2} \neq \mathcal{P}_{\mathcal{G}''_2}$, we can deduce that the filtration surfaces $\mathcal{R}_{\mathcal{G}'}$ and $\mathcal{R}_{\mathcal{G}''}$ are not equal. However, $\mathcal{T}_{\mathcal{G}'} = ((1,1,1),(1,2,1),(2,2,1),(2,2,1),(2,1,2)) = \mathcal{T}_{\mathcal{G}''}$.

□

**Proposition 1.** *Filtration surfaces are strictly more expressive than filtration trajectories.*

*Proof.* This is a direct consequence of the previous two lemmas. □

While we have proven that filtration surfaces are strictly more expressive than filtration trajectories, the latter representation is significantly more compact. This would result in an accelerated training process and a notable gain in terms of inference speed, however we leave experimental evaluation of this approach for future work.

# 5 Experiments

## 5.1 Experimental Setup

To evaluate the performance of our proposed methods, we conduct experiments on both synthetic and real-world datasets. All experiments were conducted on a cluster, on which 8x Intel Xeon 6134 CPUs and 64 GB of RAM were allocated for each run. We vectorize our filtration surfaces according to section 4.2.1 and use a random forest with 1000 trees and without a maximum depth as our classifier. We run 10 iterations of stratified 10-fold cross validation and report the mean and standard deviation of the accuracies.

## 5.2 Synthetic Datasets

We generate three synthetic datasets, each with 100 dynamic graphs, two node labels, five starting nodes, ten timesteps per dynamic graph and two edges added per timestep. The dynamic graph is assigned the label 1 if there are more nodes of class 1 in the final graph than of class 0, and the label 0 otherwise. The first dataset is generated by creating random graphs according to the the Barabási-Albert model [1], the second by creating random graphs according to the Erdős-Rényi model [9], and the third by creating a random starting graph via the Barabási-Albert model, and then adding edges according to the preferential attachment mechanism [2]. Edge weights are assigned randomly in the range $[1, 10]$. Our method, the node label histogram filtration surface using native edge weights (FS-NW), is able to classify almost all dynamic graphs correctly across all three datasets, as shown in Table 5.1.

## 5.3 Real-World Datasets

We evaluate our method on five real-world datasets, namely the MIT Reality Mining dataset [7], the Highschool and Infectious datasets from the *SocioPatterns* project [32], the Tumblr dataset – a subset of the Memetracker dataset [16] –, as well as a subset of the Dblp dataset [17]. All of our datasets can be found in the *TUDataset* collection [22].

| **Method** | **Dataset** | | |
|---|---|---|---|
| | *Erdős-Rényi (R)* | *Barabási-Albert (R)* | *Barabási-Albert (PA)* |
| FS-NW | 100 $\pm 0.0$ | 100 $\pm 0.0$ | 99.10 $\pm 0.54$ |

Table 5.1: Results of the experiments on the synthetic datasets. R stands for random, PA stands for preferential attachment. FS-NW denotes the filtration surfaces using native edge weights.

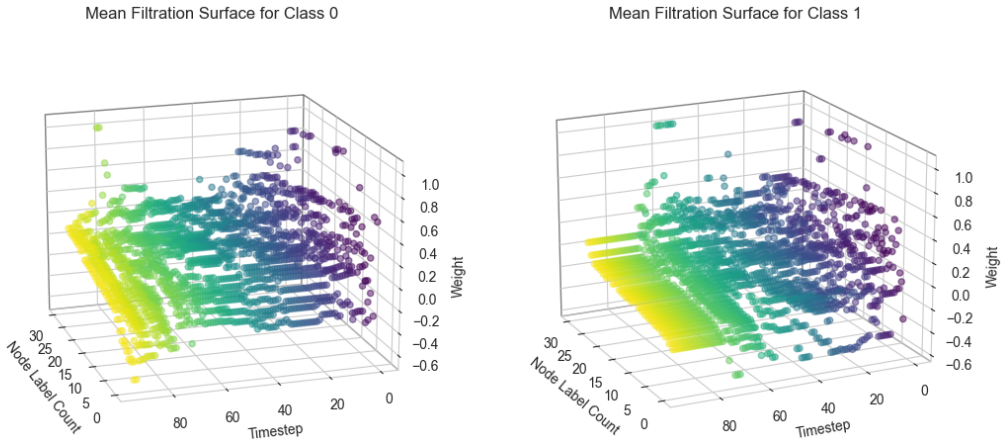



Figure 5.1: Visualisation of mean node label histogram filtration surfaces of node 0 of the Highschool dataset. The left figure shows a scatter plot of the mean filtration surface for dynamic graphs of the first class, the right figure does so for the second class. Both surfaces are taken from the first task. The mean filtration surface is calculated by individually averaging all filtration surface dimensions across all dynamic graphs of a given class. While we could have depicted the surface by forward-filling within and in-between filtration curves, we chose to use a scatter plot to avoid giving the appearance of density where it does not exist.

The datasets are obtained by generating induced subgraphs via BFS runs from each vertex. Afterwards a dissemination process is simulated on each subgraph according to the *susceptible-infected* (SI) model [6]. The SI model is a simple epidemiological model, which describes the spread of a disease in a population of susceptible and infected individuals. At the beginning of the simulation, a random node is chosen as the starting node and labeled as infected. In each timestep, each infected node infects each of its susceptible neighbours with probability $p$. In our case, the simulation stops when half of the nodes are infected. The dataset statistics are shown in Table 5.2.

For our first classification task, nodes of half of the dataset of dynamic graphs are labeled by simulating the SI model with $p = 0.5$, and nodes of the other half are labeled randomly. Former

| **Properties** | **Dataset** | | | | |
|---|---|---|---|---|---|
| | *MIT* | *Highschool* | *Infectious* | *Tumblr* | *Dblp* |
| Size | 97 | 180 | 200 | 373 | 755 |
| ∅ \|V\| | 20 | 52.3 | 50 | 53.1 | 52.9 |
| min \|E\| | 126 | 286 | 218 | 96 | 206 |
| max \|E\| | 3 363 | 517 | 505 | 190 | 225 |
| ∅ \|E\| | 702.8 | 262.4 | 220.4 | 98.7 | 156.8 |
| ∅ max $d(v)$ | 680.7 | 92.5 | 43.8 | 24.4 | 26.4 |

Table 5.2: Statistics of the datasets (from [25]).

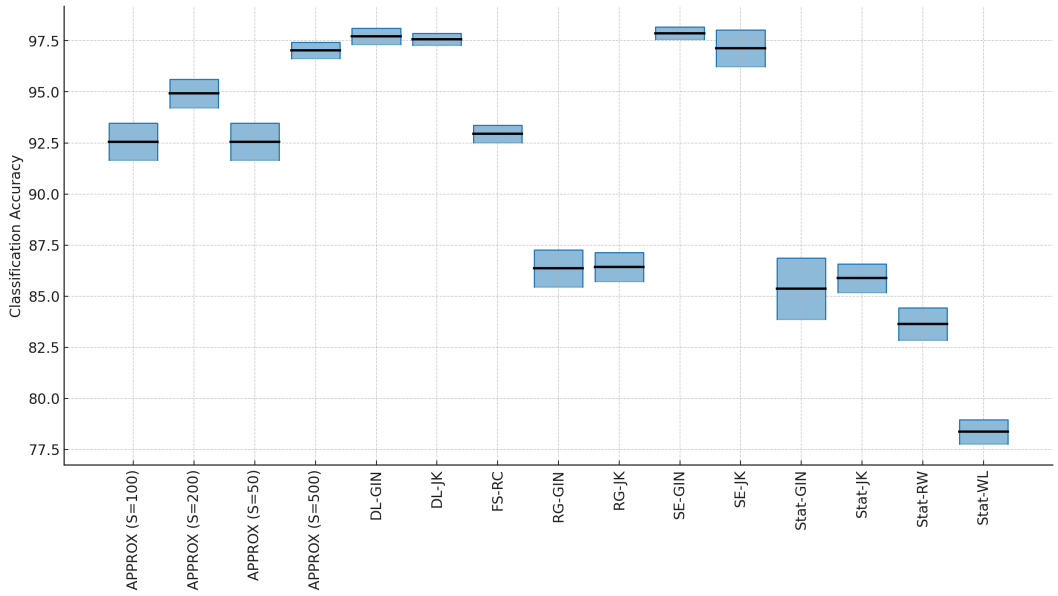



Figure 5.2: Boxplot of the first classification task on the Dblp dataset. The boxes depict the standard deviation, while the line inside the box represents the mean. We omit the temporal kernel methods from the boxplot, as they are not comparable to the other methods in terms of scalability.

dynamic graphs are assigned class 0, while the latter are of class 1. For our second classification task, nodes of both dynamic graph classes are labeled using the SI model with $p = 0.2$ for dynamic graphs of class 0 and $p = 0.8$ for the others. The results of the first and second classification task are shown in Table 5.3 and Table 5.4, respectively. We used node label histogram filtration surfaces with Ricci curvature as our edge weight function (FS-RC), since the dynamic graphs of the datasets do not have native edge weights. Figure 5.2 shows a box plot of our approach and comparable methods on the Dblp dataset. Methods that start with *Stat-* are static graph classification techniques that were applied to dynamic graphs by converting them to spatiotemporal versions. *RG*, *DL* and *SE* [25] are three approaches for converting dynamic graphs to static graphs in a way that maximizes the preservation of temporal information. The *APPROX* methods [25] are stochastic variants of *DL* with provable approximation guarantees. *S* in *APPROX(S=k)* denotes the number of temporal walks used for the approximation.

The results show that our method is able to outperform the static methods in the first classification task on all but the Tumblr dataset and exhibits comparable or higher accuracies than Graph Isomorphism Networks (GIN) [46] and Jumping Knowledge Networks (JK) [47] on the *RG* representation. One some datasets, a filtration surface is also comparable in performance to non-kernel-based methods on the *DL* and *SE* representations, as well as the *APPROX* methods, but on other datasets, it falls short of their performance. Only the temporal variants of the random walk kernel (RW) [33] and the Weisfeiler-Lehman subtree kernel (WL) [30] consistently outperform our approach, both of which have scalability issues as discussed in section 3.4. Overall, our method exhibits a significantly lower standard deviation than any other method.

| | Method | Dataset | | | | |
|---|---|---|---|---|---|---|
| | | *MIT* | *Highschool* | *Infectious* | *Tumblr* | *Dblp* |
| Static | *Stat-RW* | 61.03 ±2.4 | 61.61 ±4.3 | 75.80 ±1.6 | 79.50 ±1.6 | 83.64 ±0.8 |
| | *Stat-WL* | 43.48 ±1.9 | 48.38 ±1.5 | 64.95 ±5.3 | 76.87 ±0.9 | 78.36 ±0.6 |
| | *Stat-GIN* | 65.20 ±4.5 | 50.77 ±5.4 | 66.05 ±3.7 | 74.46 ±2.1 | 85.37 ±1.5 |
| | *Stat-JK* | OOM | 49.17 ±3.7 | 53.60 ±3.8 | 71.69 ±1.7 | 85.88 ±0.7 |
| Dynamic | *RG-RW* | 61.31 ±2.7 | 90.16 ±1.0 | 89.30 ±1.0 | 74.99 ±1.9 | 90.60 ±1.0 |
| | *RG-WL* | 81.88 ±1.1 | 89.88 ±0.9 | 91.75 ±1.0 | 70.50 ±1.0 | 90.45 ±0.5 |
| | *RG-GIN* | 50.65 ±4.2 | 51.11 ±2.5 | 58.20 ±4.0 | 72.63 ±1.8 | 86.36 ±0.9 |
| | *RG-JK* | 50.74 ±3.1 | 50.83 ±4.9 | 47.85 ±2.7 | 69.14 ±3.6 | 86.43 ±0.7 |
| | *DL-RW* | 92.91 ±0.9 | 98.33 ±0.7 | 97.05 ±0.8 | 94.64 ±0.5 | 98.16 ±0.1 |
| | *DL-WL* | 90.67 ±1.6 | 98.88 ±0.4 | 97.35 ±1.5 | 94.05 ±0.9 | 98.56 ±0.3 |
| | *DL-GIN* | OOM | 88.67 ±2.1 | 92.85 ±1.7 | 90.39 ±1.4 | 97.72 ±0.4 |
| | *DL-JK* | OOM | 86.22 ±2.6 | 91.55 ±2.3 | 89.30 ±1.5 | 97.57 ±0.3 |
| | *SE-RW* | 88.56 ±1.0 | 96.89 ±1.2 | 97.60 ±0.6 | 93.97 ±0.9 | 98.65 ±0.3 |
| | *SE-WL* | 87.31 ±1.9 | 96.72 ±0.7 | 94.45 ±1.1 | 93.51 ±0.6 | 97.38 ±0.2 |
| | *SE-GIN* | 75.98 ±3.7 | 92.28 ±1.2 | 93.10 ±1.9 | 92.78 ±1.1 | 97.87 ±0.3 |
| | *SE-JK* | 75.37 ±3.6 | 92.33 ±2.7 | 93.50 ±1.9 | 92.30 ±0.9 | 97.14 ±0.9 |
| | *APPROX (S=50)* | OOM | 81.66 ±1.7 | 84.55 ±1.6 | 86.92 ±1.2 | 92.56 ±0.9 |
| | *APPROX (S=100)* | 81.88 ±1.0 | 81.66 ±1.7 | 84.55 ±1.6 | 86.92 ±1.2 | 92.56 ±0.9 |
| | *APPROX (S=200)* | 83.69 ±3.6 | 86.11 ±1.2 | 89.35 ±1.6 | 90.62 ±0.6 | 94.92 ±0.7 |
| | *APPROX (S=500)* | 84.26 ±3.3 | 91.05 ±6.4 | 91.85 ±1.7 | 92.73 ±0.9 | 97.03 ±0.4 |
| | *FS-RC* | 58.88 ±1.97 | 82.56 ±0.83 | 83.55 ±0.99 | 68.16 ±1.31 | 92.94 ±0.42 |

Table 5.3: Table showing classification accuracies in percent and standard deviation for the first classification task. OOM means out of memory. The accuracies of the comparison methods are taken from [24]. FS-RC denotes filtration surfaces with Ollivier-Ricci curvature as the edge weight function.

| | Method | Dataset | | | | |
|---|---|---|---|---|---|---|
| | | *MIT* | *Highschool* | *Infectious* | *Tumblr* | *Dblp* |
| Static | *Stat-RW* | 56.84 ±2.6 | 62.83 ±2.9 | 63.05 ±1.4 | 65.26 ±1.9 | 61.17 ±0.9 |
| | *Stat-WL* | 42.42 ±3.9 | 60.83 ±3.2 | 63.60 ±1.3 | 68.31 ±1.5 | 63.11 ±0.9 |
| | *Stat-GIN* | 55.60 ±11.0 | 54.05 ±4.7 | 55.25 ±3.3 | 64.99 ±1.1 | 61.92 ±1.2 |
| | *Stat-JK* | OOM | 53.16 ±3.2 | 53.00 ±2.9 | 65.42 ±2.8 | 61.34 ±1.1 |
| Dynamic | *RG-RW* | 58.03 ±3.7 | 77.33 ±2.4 | 72.05 ±2.2 | 68.48 ±1.5 | 63.24 ±1.2 |
| | *RG-WL* | 66.81 ±2.0 | 82.78 ±1.3 | 77.40 ±1.2 | 68.25 ±1.2 | 66.16 ±0.5 |
| | *RG-GIN* | 53.80 ±16.0 | 53.61 ±3.5 | 51.80 ±4.2 | 64.70 ±2.4 | 60.24 ±1.8 |
| | *RG-JK* | 51.80 ±9.7 | 54.61 ±2.9 | 52.60 ±3.2 | 65.50 ±2.6 | 61.00 ±1.1 |
| | *DL-RW* | 82.64 ±2.1 | 91.44 ±1.1 | 87.35 ±1.3 | 76.51 ±0.5 | 81.79 ±0.9 |
| | *DL-WL* | 40.87 ±3.6 | 87.11 ±1.7 | 77.55 ±2.0 | 78.69 ±0.8 | 74.47 ±1.1 |
| | *DL-GIN* | OOM | 89.11 ±2.0 | 80.60 ±2.2 | 75.45 ±2.4 | 80.05 ±1.1 |
| | *DL-JK* | OOM | 85.00 ±2.8 | 75.70 ±3.5 | 73.10 ±1.8 | 79.98 ±1.3 |
| | *SE-RW* | 51.03 ±5.1 | 90.77 ±1.1 | 83.60 ±1.1 | 77.09 ±1.0 | 83.31 ±1.0 |
| | *SE-WL* | 46.52 ±3.9 | 91.55 ±0.9 | 79.60 ±1.5 | 78.64 ±1.4 | 81.24 ±0.6 |
| | *SE-GIN* | 51.40 ±11.1 | 85.88 ±2.1 | 75.05 ±3.4 | 73.23 ±1.7 | 80.72 ±1.1 |
| | *SE-JK* | 51.40 ±10.9 | 82.44 ±2.0 | 74.25 ±2.4 | 74.55 ±1.4 | 81.18 ±1.0 |
| | *APPROX (S=50)* | 55.81 ±3.2 | 77.94 ±1.8 | 71.70 ±2.2 | 72.96 ±1.4 | 68.70 ±0.8 |
| | *APPROX (S=100)* | 59.24 ±5.5 | 83.56 ±1.1 | 76.25 ±2.5 | 73.03 ±2.3 | 72.50 ±0.7 |
| | *APPROX (S=250)* | 59.48 ±3.9 | 88.56 ±1.7 | 78.75 ±3.0 | 75.47 ±1.2 | 74.44 ±0.9 |
| | *FS-RC* | 53.34 ±1.78 | 66.67 ±0.93 | 67.45 ±1.06 | 63.56 ±1.04 | 63.27 ±0.78 |

Table 5.4: Table showing classification accuracies in percent and standard deviation for the second classification task. OOM means out of memory.

# 6 Conclusion

We extend prior work on *filtration curves* [23] to the dynamic setting and introduce *filtration surfaces*, a method for classifying discrete-time dynamic graphs consisting of labeled or unlabeled graphs. Experimentally, filtration surfaces outperform static methods applied to spatiotemporal graphs. Moreover, they are comparable in performance to non-kernel-based methods and approximate kernel methods applied to static graphs, which were obtained by transforming dynamic graphs using techniques described in [25].

For even better scalability, we present an alternative method, which we term *filtration trajectories*, and prove that its expressivity is upper-bounded by that of filtration surfaces. Since the conversion approaches described in [25] lead to a blow-up of the size of the static graph, we argue that our methods pose scalable and attractive alternatives for dynamic graph classification.

There are several promising future directions to explore, such as learning the interpolation function between curves of filtration surfaces in cases where the dynamic graph is obtained by sampling from an underlying "ground truth" dynamic graph, learning an extrapolation function by masking parts of the filtration surface, and experimentally evaluating the efficacy of filtration trajectories on dynamic graphs. Conducting an extensive runtime analysis of filtration surfaces in comparison with other methods for dynamic graph classification would also constitute a valuable future contribution. Lastly, we believe that the field of dynamic graph classification would benefit from a more thorough investigation of novel modelling approaches for dynamic graphs, such as the one proposed in [3].

# Bibliography


1. R. Albert and A.-L. Barabási. "Statistical mechanics of complex networks". *Reviews of modern physics* 74:1, 2002, p. 47.
2. A.-L. Barabási and R. Albert. "Emergence of scaling in random networks". *science* 286:5439, 1999, pp. 509–512.
3. W. Bernardoni, R. Cardona, J. Cleveland, J. Curry, R. Green, B. Heller, A. Hylton, T. Lam, and R. Kassouf-Short. *Algebraic and Geometric Models for Space Networking*. 2023. arXiv: 2304.01150 [math.AT].
4. M. Carrière, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. "Perslay: A neural network layer for persistence diagrams and new graph topological signatures". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2786–2796.
5. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". *arXiv preprint arXiv:1810.04805*, 2018.
6. O. Diekmann and J. A. P. Heesterbeek. *Mathematical epidemiology of infectious diseases: model building, analysis and interpretation*. Vol. 5. John Wiley & Sons, 2000.
7. N. Eagle and A. Pentland. "Reality mining: sensing complex social systems". *Personal and ubiquitous computing* 10, 2006, pp. 255–268.
8. H. Edelsbrunner and J. L. Harer. *Computational topology: an introduction*. American Mathematical Society, 2022, pp. 178–185.
9. P. ERDdS and A. R&wi. "On random graphs I". *Publ. math. debrecen* 6:290-297, 1959, p. 18.
10. K. Feng, C. Li, X. Zhang, and J. ZHOU. "Towards Open Temporal Graph Neural Networks". In: *The Eleventh International Conference on Learning Representations*. 2022.
11. W. Hamilton, Z. Ying, and J. Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.
12. M. Jin, Y.-F. Li, and S. Pan. "Neural Temporal Walks: Motif-Aware Representation Learning on Continuous-Time Dynamic Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 19874–19886. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/7dadc855cef7494d5d956a8d28add871-Paper-Conference.pdf.

13. L. V. Kantorovich. "Mathematical methods of organizing and planning production". *Management science* 6:4, 1960, pp. 366–422.

14. B.-H. Kim, J. C. Ye, and J.-J. Kim. "Learning dynamic graph representation of brain connectome with spatio-temporal attention". *Advances in Neural Information Processing Systems* 34, 2021, pp. 4314–4327.

15. N. M. Kriege, F. D. Johansson, and C. Morris. "A survey on graph kernels". *Applied Network Science* 5:1, 2020.

16. J. Leskovec, L. Backstrom, and J. Kleinberg. "Meme-tracking and the dynamics of the news cycle". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 497–506.

17. M. Ley. "The DBLP computer science bibliography: Evolution, research issues, perspectives". In: *International symposium on string processing and information retrieval*. Springer. 2002, pp. 1–10.

18. Z. Li, K. Fujiwara, and G. Tanaka. *Dynamical Graph Echo State Networks with Snapshot Merging for Dissemination Process Classification*. 2023. arXiv: 2307.01237 [cs.LG].

19. F. Manessi, A. Rozza, and M. Manzo. "Dynamic graph convolutional networks". *Pattern Recognition* 97, 2020, p. 107000. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2019.107000. URL: https://www.sciencedirect.com/science/article/pii/S0031320319303036.

20. N. Masuda and R. Lambiotte. *A Guide to Temporal Networks*. World Scientific, 2016. ISBN: 978-1-78634-114-3. DOI: 10.1142/q0268.

21. A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. "Automating the construction of internet portals with machine learning". *Information Retrieval* 3, 2000, pp. 127–163.

22. C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. *TUDataset: A collection of benchmark datasets for learning with graphs*. 2020. arXiv: 2007.08663 [cs.LG].

23. L. O'Bray, B. Rieck, and K. Borgwardt. "Filtration Curves for Graph Representation". In: *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. Association for Computing Machinery, New York, NY, USA, 2021. DOI: 10.1145/3447548.3467442. In press.

24. L. Oettershagen. "Temporal Graph Algorithms". PhD thesis. Universitäts-und Landesbibliothek Bonn, 2022.

25. L. Oettershagen, N. M. Kriege, C. Morris, and P. Mutzel. "Temporal Graph Kernels for Classifying Dissemination Processes". In: *Proceedings of the 2020 SIAM International Conference on Data Mining (SDM)*, pp. 496–504. DOI: 10.1137/1.9781611976236.56. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976236.56. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976236.56.

26. Y. Ollivier. "Ricci curvature of Markov chains on metric spaces". *Journal of Functional Analysis* 256:3, 2009, pp. 810–864. ISSN: 0022-1236. DOI: https://doi.org/10.1016/j.jfa.2008.11.001. URL: https://www.sciencedirect.com/science/article/pii/S002212360800493X.

27. R. C. Read and D. G. Corneil. "The graph isomorphism disease". *Journal of graph theory* 1:4, 1977, pp. 339–363.

28. E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. *Temporal Graph Networks for Deep Learning on Dynamic Graphs*. 2020. arXiv: 2006.10637 [cs.LG].

29. B. Rozemberczki, C. Allen, and R. Sarkar. "Multi-Scale attributed node embedding". *Journal of Complex Networks* 9:2, 2021, cnab014. ISSN: 2051-1329. DOI: 10.1093/comnet/cnab014. eprint: https://academic.oup.com/comnet/article-pdf/9/2/cnab014/40435146/cnab014.pdf. URL: https://doi.org/10.1093/comnet/cnab014.

30. N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research* 12:9, 2011.

31. J. Skarding, B. Gabrys, and K. Musial. "Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey". *IEEE Access* 9, 2021, pp. 79143–79168. DOI: 10.1109/ACCESS.2021.3082932.

32. *Sociopatterns*. http://www.sociopatterns.org. Accessed: August 8, 2023.

33. M. Sugiyama and K. Borgwardt. "Halting in random walk kernels". *Advances in neural information processing systems* 28, 2015.

34. J. Sun, M. Ovsjanikov, and L. Guibas. "A concise and provably informative multi-scale signature based on heat diffusion". In: *Computer graphics forum*. Vol. 28. 5. Wiley Online Library. 2009, pp. 1383–1392.

35. A. Taheri, K. Gimpel, and T. Berger-Wolf. "Learning to Represent the Evolution of Dynamic Graphs with Recurrent Models". In: *Companion Proceedings of The 2019 World Wide Web Conference*. WWW '19. Association for Computing Machinery, San Francisco, USA, 2019, pp. 301–307. ISBN: 9781450366755. DOI: 10.1145/3308560.3316581. URL: https://doi.org/10.1145/3308560.3316581.

36. M. Togninalli, E. Ghisu, F. Llinares-López, B. Rieck, and K. Borgwardt. "Wasserstein Weisfeiler-Lehman Graph Kernels". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/73fed7fd472e502d8908794430511f4d-Paper.pdf.

37. D. Tortorella, A. Micheli, et al. "Dynamic Graph Echo State Networks". In: *ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. i6doc. 2021, pp. 99–104.

38. R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha. "DyRep: Learning Representations over Dynamic Graphs". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyePrhR5KX.

39. R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha. "Representation Learning over Dynamic Graphs". *CoRR* abs/1803.04051, 2018. arXiv: 1803.04051. URL: http://arxiv.org/abs/1803.04051.

40. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". *Advances in neural information processing systems* 30, 2017.

41. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. "Graph Attention Networks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

42. S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, and J. Yang. "Multiscale Dynamic Graph Convolutional Network for Hyperspectral Image Classification". *IEEE Transactions on Geoscience and Remote Sensing* 58:5, 2020, pp. 3162–3177. DOI: 10.1109/TGRS.2019.2949180.

43. H. Wang, J. Wu, X. Zhu, Y. Chen, and C. Zhang. "Time-variant graph classification". *IEEE Transactions on systems, man, and cybernetics: systems* 50:8, 2018, pp. 2883–2896.

44. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds". *ACM Trans. Graph.* 38:5, 2019. ISSN: 0730-0301. DOI: 10.1145/3326362. URL: https://doi.org/10.1145/3326362.

45. K. Xu, W. Hu, J. Leskovec, and S. Jegelka. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG].

46. K. Xu, W. Hu, J. Leskovec, and S. Jegelka. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

47. K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. "Representation learning on graphs with jumping knowledge networks". In: *International conference on machine learning*. PMLR. 2018, pp. 5453–5462.

48. D. Ye, H. Jiang, Y. Jiang, and H. Li. "Stable distance of persistent homology for dynamic graph comparison". *Knowledge-Based Systems*, 2023, p. 110855. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2023.110855. URL: https://www.sciencedirect.com/science/article/pii/S0950705123006056.

49. L. Ye and E. Keogh. "Time Series Shapelets: A New Primitive for Data Mining". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. Association for Computing Machinery, Paris, France, 2009, pp. 947–956. ISBN: 9781605584959. DOI: 10.1145/1557019.1557122. URL: https://doi.org/10.1145/1557019.1557122.

50. S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. "Graph Transformer Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

51. Z. Zhang, X. Wang, Z. Zhang, H. Li, Z. Qin, and W. Zhu. "Dynamic Graph Neural Networks Under Spatio-Temporal Distribution Shift". In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. 2022. URL: https://openreview.net/forum?id=1tIUqrUuJxx.

# Appendices

# Additional Figures

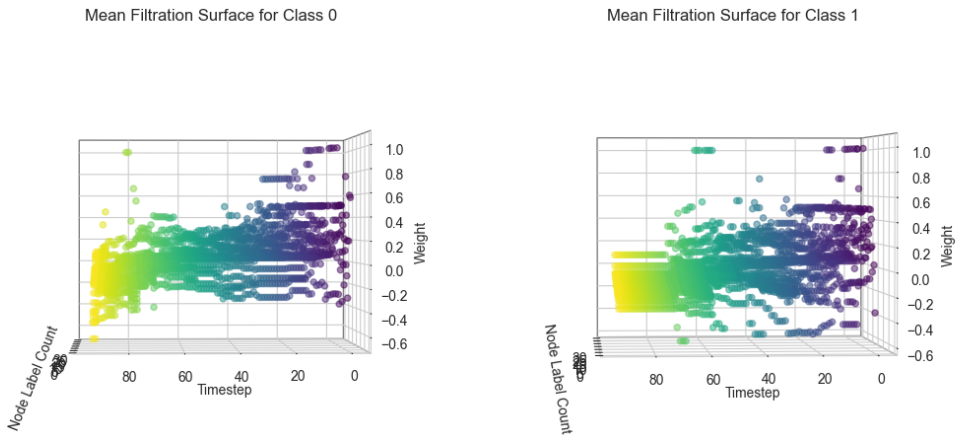



Figure 1: Frontal view of mean node label histogram filtration surfaces of node 0 of the Highschool dataset.

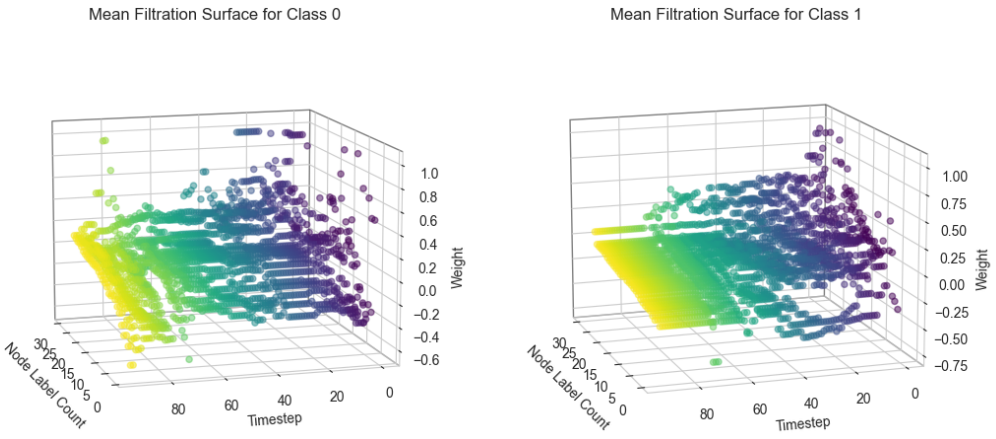



Figure 2: Visualisation of mean node label histogram filtration surfaces of node 1 of the Highschool dataset.

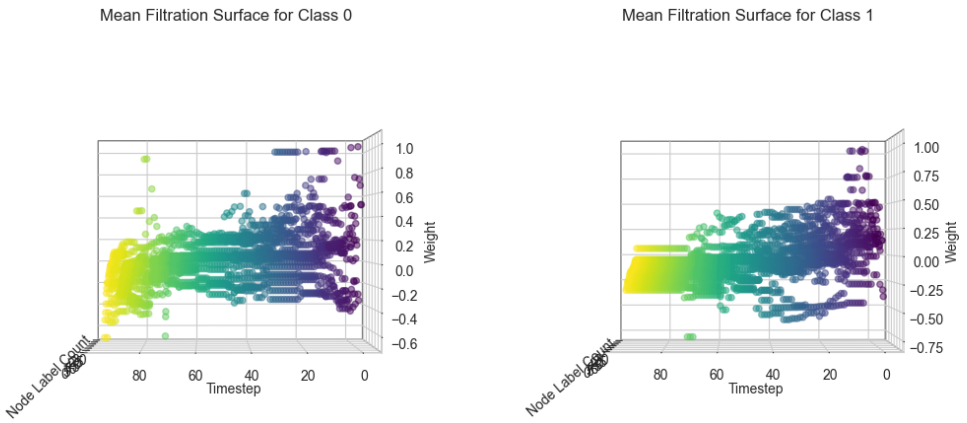



Figure 3: Frontal view of mean node label histogram filtration surfaces of node 1 of the Highschool dataset.

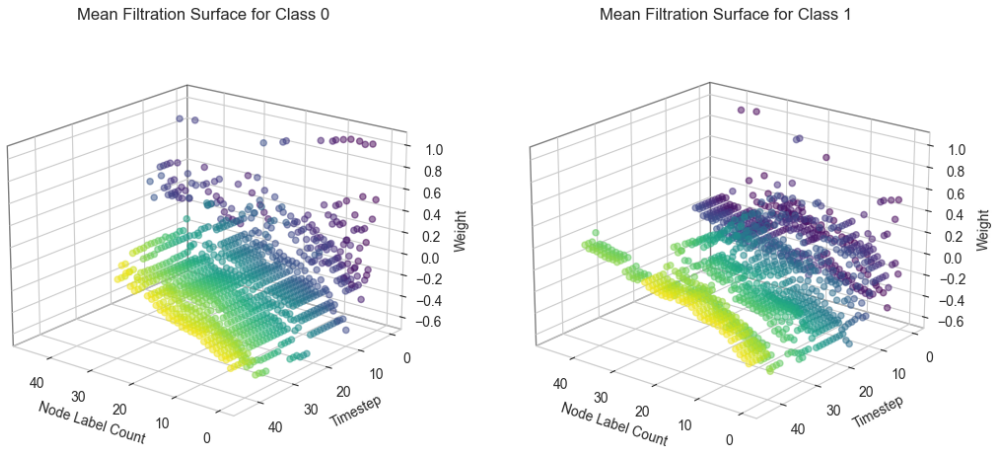



Figure 4: Visualisation of mean node label histogram filtration surfaces of node 0 of the Infectious dataset.

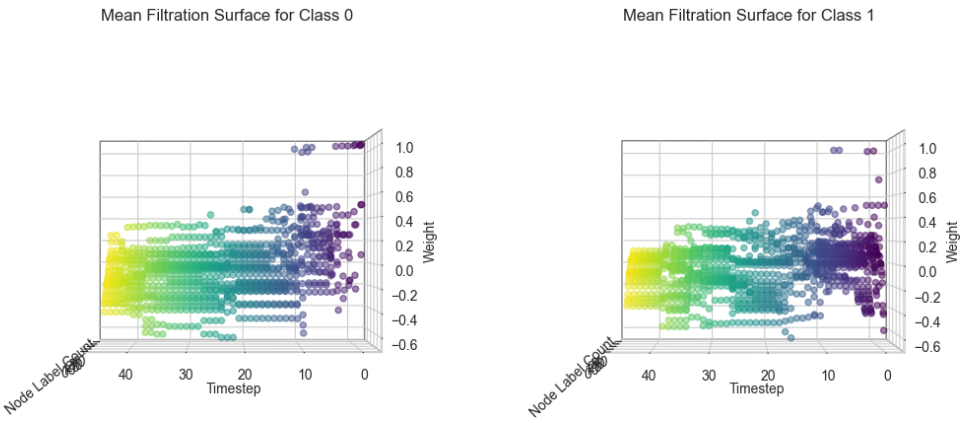



Figure 5: Frontal view of mean node label histogram filtration surfaces of node 0 of the Infectious dataset.

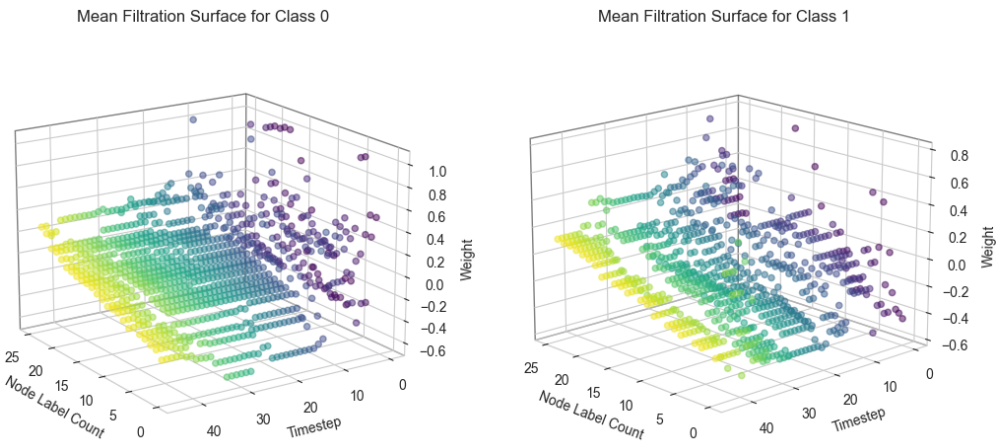



Figure 6: Visualisation of mean node label histogram filtration surfaces of node 1 of the Infectious dataset.

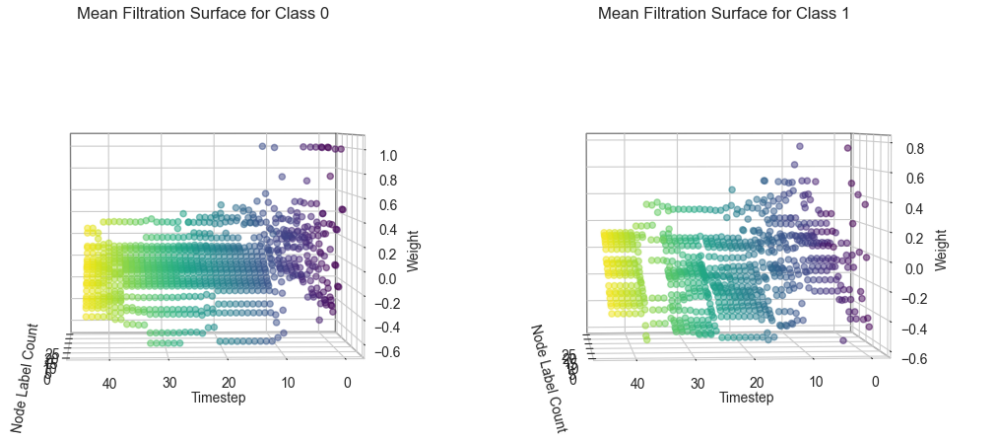



Figure 7: Frontal view of mean node label histogram filtration surfaces of node 1 of the Infectious dataset.

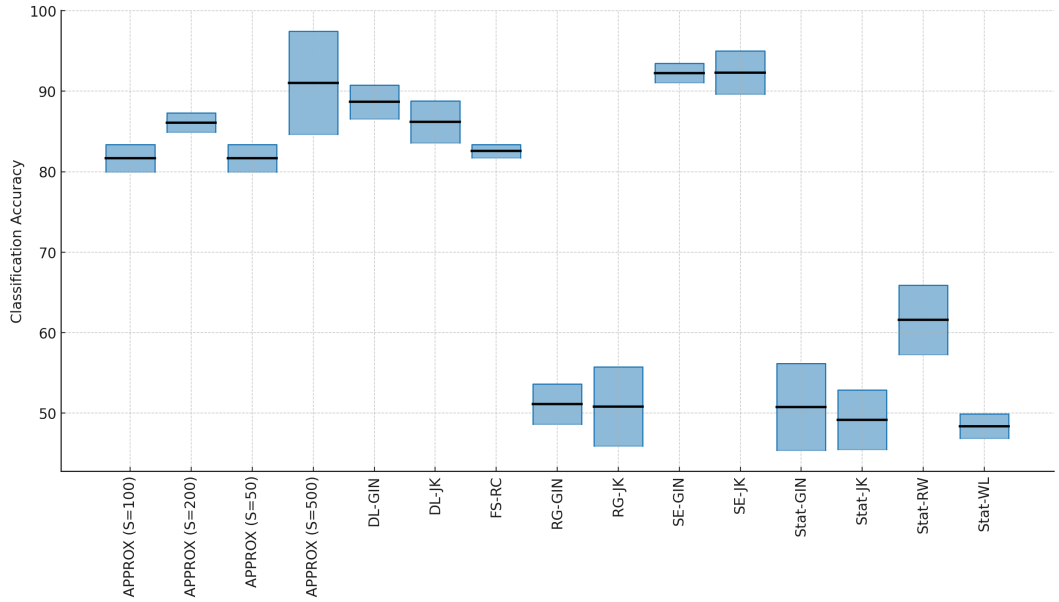



Figure 8: Boxplot of the first classification task on the Highschool dataset.

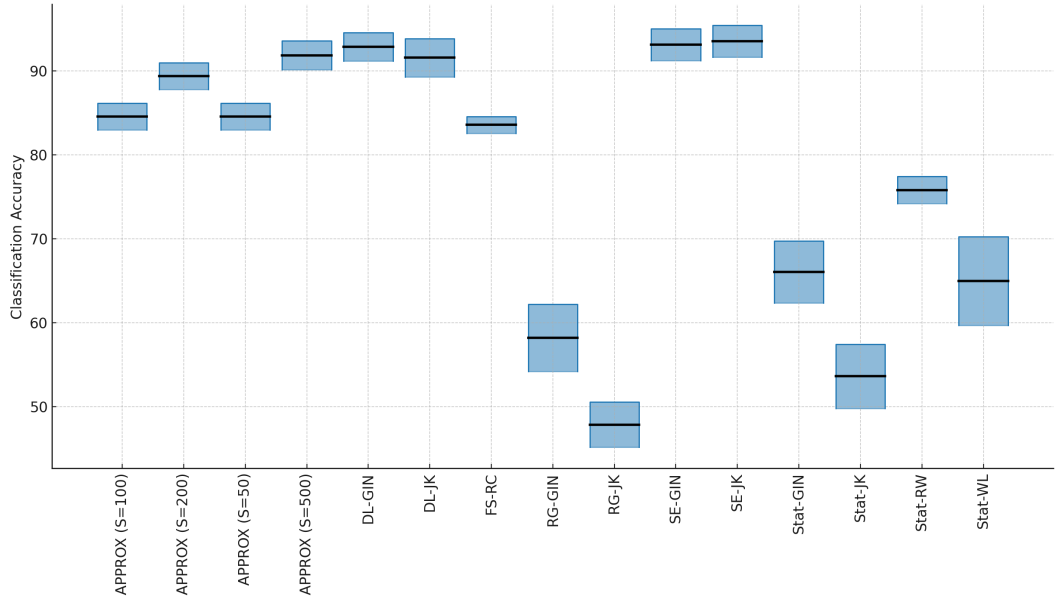



Figure 9: Boxplot of the first classification task on the Infectious dataset.

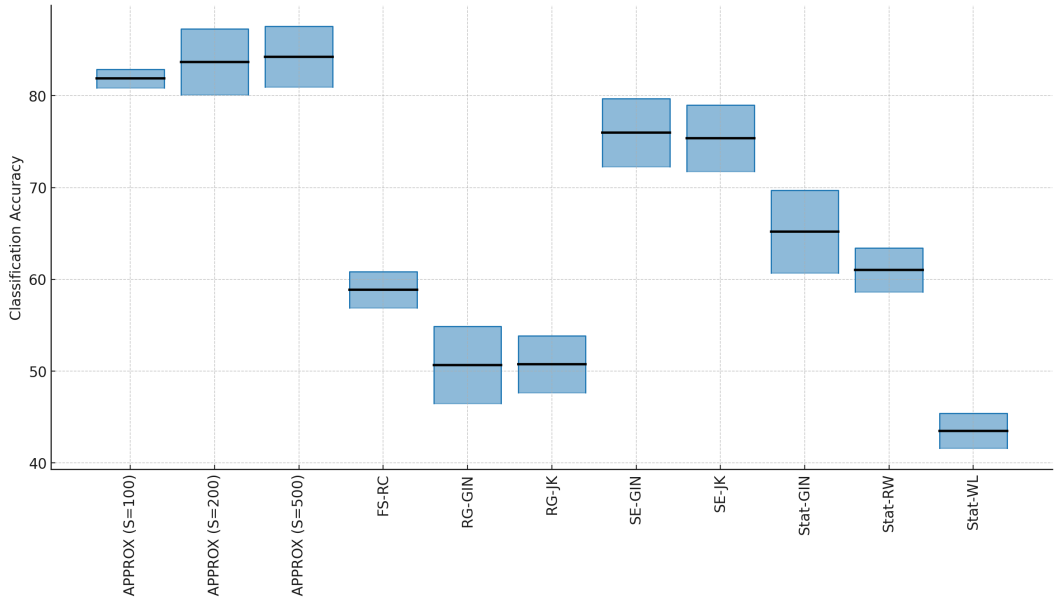



Figure 10: Boxplot of the first classification task on the MIT dataset.

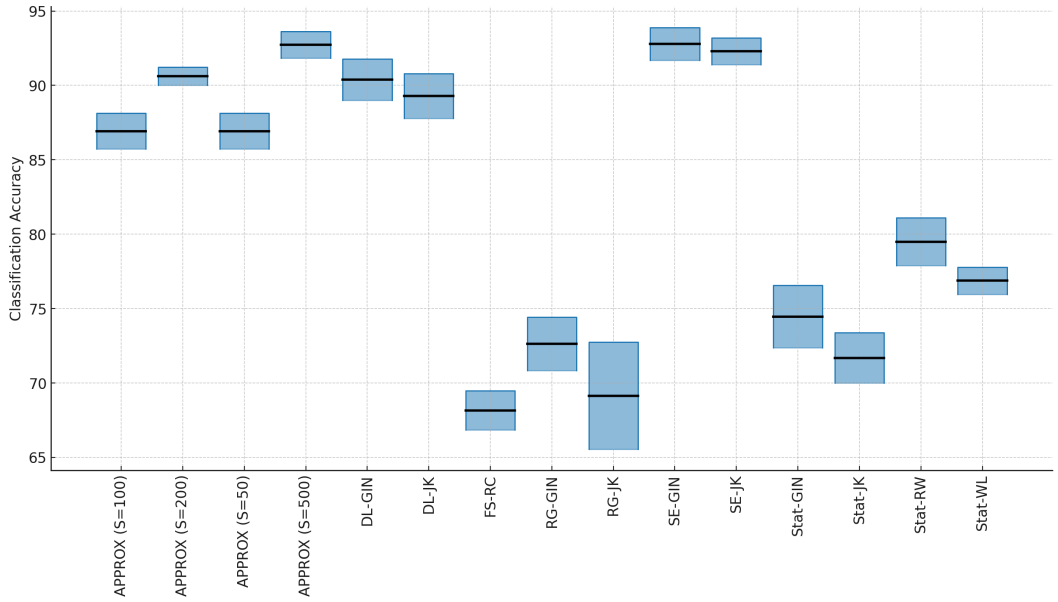



Figure 11: Boxplot of the first classification task on the Tumblr dataset.

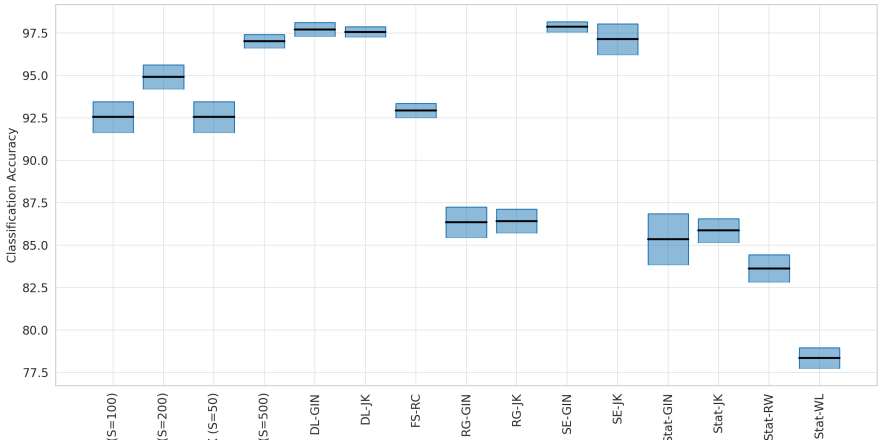



Figure 12: Boxplot of the second classification task on the Dblp dataset.

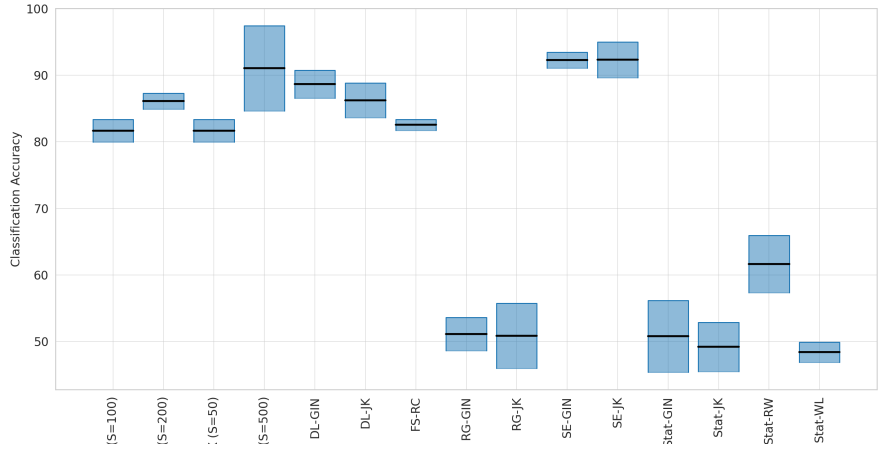



Figure 13: Boxplot of the second classification task on the Highschool dataset.

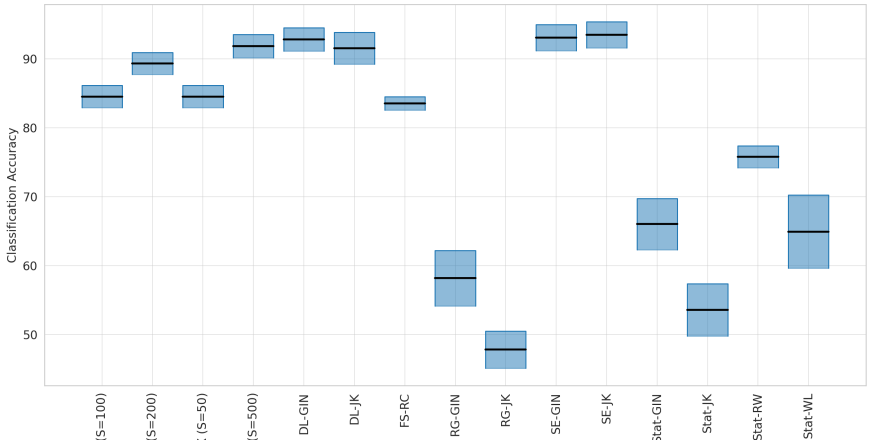



Figure 14: Boxplot of the second classification task on the Infectious dataset.

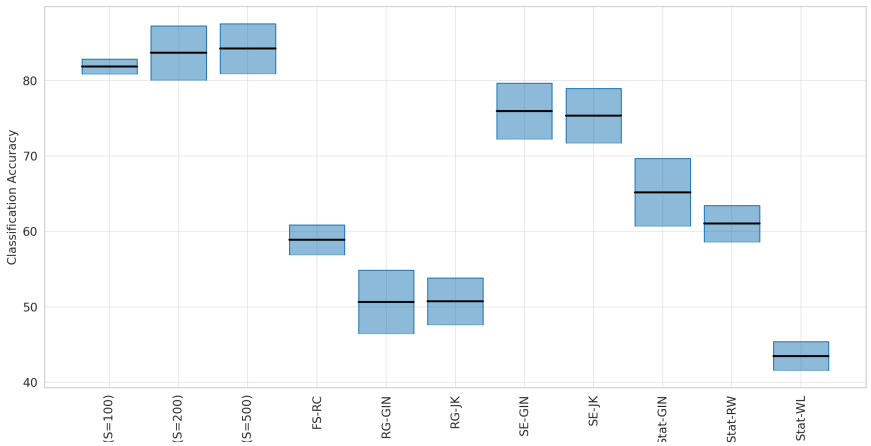



Figure 15: Boxplot of the second classification task on the MIT dataset.

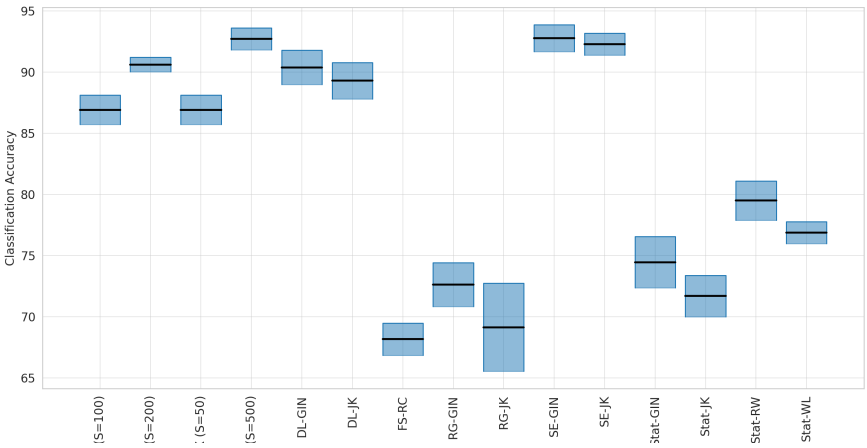



Figure 16: Boxplot of the second classification task on the Tumblr dataset.