Technische Universität Berlin
Fakultät IV: Elektrotechnik und Informatik
Institut für Softwaretechnik und Theoretische Informatik
Algorithmik und Komplexitätstheorie (AKT)

Methods for Comparing Temporal Graphs: An empirical Study

Masterarbeit

von Jonas Schulte-Mattler

zur Erlangung des Grades „Master of Science“ (M. Sc.)
im Studiengang Computer Science (Informatik)

Erstgutachter:	Prof. Dr. Mathias Weller
Zweitgutachter:	Prof. Dr. Stefan Schmid
Betreuer:	Dr. Vincent Froese, Malte Renken

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den	24.10.2023	[illegible]
	Datum	Unterschrift

Abstract

Dynamic networks or temporal graphs that consist of objects whose relationships change over time appear in many real-word systems. Solving machine learning tasks as pattern recognition, clustering, or classification of temporal graphs requires a proximity measure between them. To this end, new concepts have been proposed to measure temporal graph proximity. This work empirically studies these methods. We consider two applications: Dissemination processes and dynamic human brain connectivity. We compare recently introduced temporal graph kernels to classify dissemination processes with another method called dynamic temporal graph warping (dtgw). In our second experiment we model dynamic functional brain connectivity as a temporal graph and evaluate temporal graph proximity methods to classify brain diseases. We found that temporal graph kernels are a more suitable proximity measure for dissemination processes than dtgw and that temporal graph comparison methods can be applied to fMRI data in order to classify brain connetomes by their functional connectivity.

Zusammenfassung

Dynamische Netzwerke oder temporale Graphen, die aus Objekten bestehen, deren Beziehungen sich über die Zeit ändern, tauchen in vielen Bereichen auf. Probleme des maschinellen Lernens wie Mustererkennung, Clustering oder Klassifizierung von temporalen Graphen benötigen ein Ähnlichkeitsmaß. Zu diesem Zweck sind neue Konzepte entwickelt worden. In dieser Arbeit werden diese Methoden empirisch evaluiert. Zwei Anwendungen werden betrachtet: Ausbreitungsprozesse und das Gehirn als dynamisches Netzwerk. Wir passen das Konzept dynamic temporal graph warping (dtgw) an temporale Graphen an, die Ausbreitungsprozesse modellieren, und verlgeichen es mit temporalen graph kernels, die eingeführt wurden mit dem Ziel, solche Prozesse zu klassifizieren. In unserem zweiten Experiment modellieren wir die dynamische funtionale Verbindungen von Hirnregionen als temporalen Graph und testen Methoden zur Ählichkeit der temporalen Graphen, um Hirnerkrankungen zu klassifizieren. Wir fanden heraus, dass temorale graph kernels ein geeigneteres Ähnlichkeitsmaß für Ausbreitungsprozesse sind als dtgw und dass Methoden zum Vergleichen von temporalen Graphen auf fMRI Daten angewandt werden könnnen, um Gehirne anhand ihrer funtionalen Konnektivität zu unterscheiden.

Contents

1 Introduction

Imagine an epidemiologist whose job is to detect possibly dangerous infection processes who observes a wave of viral diseases. He wonders how that viral infection spread behaves compared to other known infection processes and observes the infections closely. Knowing when people were infected and who was in contact with whom at each day he draws a temporal network and marks the infections – Figure 1.1. Now he faces the task to compare the resulting binary vertex labeled temporal graph with other temporal graphs to classify the virus spreading behavior.

Therefore, a *proximity measure for temporal graphs*, graphs whose edges can appear and disappear over time, is necessary. Apart from modeling disease spreading, or in general dissemination processes (e.g. information or influence), temporal graphs arise in various domains such as social networks, transportation systems, the internet, biological processes, and functional brain connectivity [Hos+22; HS12; Li+17; Oet+b; TBF17]. They provide a powerful tool to model the dynamic nature of real-world systems by capturing relationships that are inherently time-dependent. Moreover, the increasing availability of high-resolution temporal (big) data and the growing significance of time-aware analysis led to a recent boom in studying temporal graphs.

An intrinsic concept in data mining to solve tasks such as classification, clustering, or pattern recognition is a proximity measure between objects that quantifies their (dis) similarity. Example data mining problems of temporal graphs include community evolution in social networks [Dak+19], motif detection and counting [PBL17], anomaly detection [ATK14], and clustering in dynamic Protein-Interaction-Networks [OY+14].

While traditional static graph theory has been widely studied, leading to the development of numerous algorithms to compare graph structures [KJM20a], temporal graph proximity is subject to new research. It calls for novel methodologies that account for the graphs' temporal evolving topology.

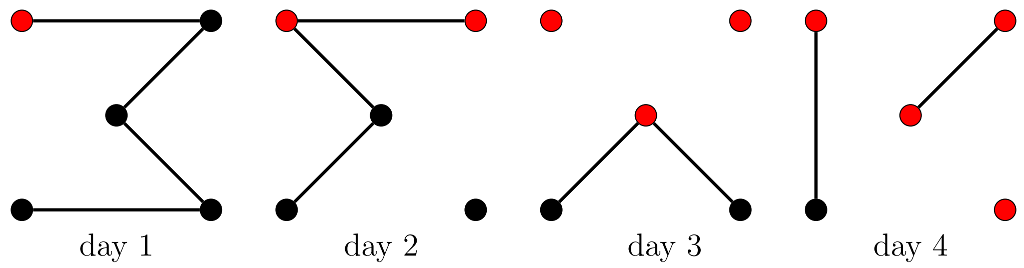


Figure 1.1: Example infection spreading process modeled by a vertex labeled temporal graph. Edges are only present at certain time steps. Infected vertices are labeled red. Infections propagate along the temporal edges.

Related work. Oettershagen et al. [Oet+b] developed kernels for temporal graphs in order to classify dissemination processes. They bring the idea of walk kernels to temporal graphs by building static graphs that capture temporal walks and apply static graph walk kernels to them. Another kernel lifts the idea of graphlets (finding correspondences between subgraphs) to the temporal setting [Oet+a].

Related concepts have been developed to make use of machine learning methods on discrete data structures as temporal graphs. To this end, vertices or networks are embedded into vectors preserving certain time-varying structural properties [Bel+20; SGR19; Tal+21]. Besides, neural networks are used to learn dynamic graph representations [Kaz+20]. These methods aim to predict node labels or links within dynamic networks.

Froese et al. [Fro+20] adapted dynamic time warping, a technique for time series comparison, to measure similarity between graph sequences. This method captures temporal variations, providing insights into how the graphs change over time.

Our contribution. This work empirically studies methods to compare temporal graphs. We present two experiments with applications in dissemination processes and brain connectome analysis.

First, we adapt the method of Froese et al. [Fro+20] to binary vertex labeled temporal graphs and compare its performance to classify dissemination processes to the temporal graph kernels recently proposed by Oettershagen et al. [Oet+a; Oet+b] on their released data sets.

Secondly, we follow a new idea to model the human brain connectome as a temporal graph [TBF17]. We present a novel approach to study and classify brain diseases using temporal graph comparison methods.

Finally, we aim to provide insights into the strengths and limitations of each comparison method and offer guidance on selecting appropriate techniques based on the characteristics of the data and research objectives.

Outline. Accordingly, this thesis is devided into the following five chapters: Chapter 2 gives notation and introduces preliminary concepts that are used in Chapter 3 to compare (temporal) graphs. After that, these methods are empirically studied considering two applications: In Chapter 4 dissemination process are compared; Chapter 5 analyses and classifies brain connectomes using temporal graph comparison methods. We give a concluding evaluation of the algorithms and look into future work in Chapter 6.

2 Preliminaries

In this chapter we introduce notation and basic definitions that are used throughout this work to compare temporal graphs. Section 2.1 gives notation of (temporal) graph theory, Section 2.2 defines basic proximity measures, and Section 2.3 introduces general concepts to match two sets of objects and time series data.

2.1 Notation

Numbers. We denote the natural numbers up to n by $[n]$, that is, $[n] := \{1, 2, \ldots, n\}$ and the natural numbers ranging from n to m by $[n, m] := \{n, \ldots, m\}$. The binary encoding of a natural number n is denoted by $\mathrm{BIN}(n)$.

Sets. For a set S, we denote the set of all size-k subsets of S by $\binom{S}{k}$.

Graph Theory. A (static) undirected *graph* $G = (V, E)$ consists of vertices V and edges $E \subseteq \binom{V}{2}$. We denote the vertex and edge set of a graph G by $V(G)$ and $E(G)$ respectively. A *labeled graph* $G = (V, E, l)$ further consists of labeled vertices and edges via a labeling $l : V \cup E \to \Sigma$ to a finite alphabet Σ.

Two vertices u and v are called *neighbors* if they are connected by an edge $\{u, v\} \in E$. The set of all neighbors of vertex v is its *neighborhood* $N(v)$.

A *walk* is a sequence of vertices $(v_1, \ldots, v_l)$ such that all consecutive vertex pairs are neighbors. We call the sequence of labels $(l(v_1), \ldots, l(v_l))$ encountered on a walk w the walk's *color* $c(w)$. Two walks are common if they have the same color. A *path* is a walk in which all vertices are distinct.

The *distance* between two vertices is the number of edges in a shortest path connecting them. The *k-neighborhood* $N_k(v)$ of vertex v are all vertices that have distance k to v.

Temporal Graph Theory. In a temporal graph edges change over discrete time steps (the vertex set is static).

Formally, we define a *temporal graph* $\mathcal{G}$ as a tuple $\mathcal{G} := (V, \mathcal{E})$ consisting of a vertex set V and *temporal edges* $\mathcal{E} \subseteq \binom{V}{2} \times T$ with $T \in \mathbb{N}$ being the *lifetime* of $\mathcal{G}$. By E_t we denote edges that are present at time step t, so $E_t := \{(\{u, v\}, t) \in \mathcal{E}\}$. The vertices V of a temporal graph $\mathcal{G}$ are denoted by $V(\mathcal{G})$, its temporal edges by $\mathcal{E}(\mathcal{G})$, and the static graph that is present at timestep t is called the *t-th layer* $\mathcal{G}_t$ of $\mathcal{G}$, so $\mathcal{G}_t = (V, E_t)$. We denote the k-neighborhood at timestep t by $N_t^k(v)$. A *(strict) temporal walk* is a sequence of vertices $(v_1, \ldots, v_l)$ such that all consecutive vertex pairs are connected via a temporal edge and the time steps at which the edges are present (strictly) increase. A *labeled* *temporal graph* is a tuple $\mathcal{G} := (V, \mathcal{E}, l)$ with a labeling $l : V \times \mathcal{E} \to \Sigma$ over the label alphabet Σ. The label of vertex v at timestep t is denoted by $l_t(v)$. Figure 1.1 shows a labeled temporal graph with a binary alphabet.

2.2 Proximity Measures

First, spaces are defined in which distances and kernels are specified as proximity measures.

Spaces. A *metric space* is a set X and a distance function $d : X \times X \to \mathbb{R}$ such that all $x, y, z \in X$ satisfy:

1. $d(x, x) = 0$,
2. $x \neq y \Rightarrow d(x, y) > 0$ (positivity),
3. $d(x, y) = d(y, x)$ (symmetry),
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

A *real inner product space* is a real vector space with an inner product $\langle \cdot, \cdot \rangle$ that takes real values. An inner product induces a vector-norm

$$||x|| = \sqrt{\langle x, x \rangle}$$

and a distance function

$$d(x, y) = ||x - y||.$$

A real *Hilbert space* is a real inner product space that is also a complete metric space with respect to the distance function induced by the inner product.

Norms and Distances. For $p \geq 1$, the L_p*-norm* of a d-dimensional vector x is

$$||x||_p := \Big(\sum_{i=1}^{d} |x_i|^p \Big)^{1/p}.$$

The *distance* derived from the L_p-norm is

$$d_p(x, y) := ||x - y||_p.$$

For $p = 1$ this is the sum of the absolute values of a vector called Taxi-, City-block or Manhattan-norm; for $p = 2$ we get the Euclidean-norm.

Kernels. Let $\mathcal{X}$ be a set of objects to compare and let the features be expressed in a real Hilbert space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle$ via a *feature map* $\phi : \mathcal{X} \to \mathcal{H}$. Then the similarity between two objects $x, y \in \mathcal{X}$ is measured by $\langle \phi(x), \phi(y) \rangle$. That inner product is induced by a positive semi-definite function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ called *kernel* if and only if $\langle \phi(x), \phi(y) \rangle = \kappa(x, y)$ for all $x, y \in \mathcal{X}$.

A (temporal) graph kernel is a kernel on the set of (temporal) graphs.

2.3 Matching Sets and Time Series

We introduce two concepts to match two distinct distinct (vertex) sets and time series. They are subsequently used to compare temporal graphs.

Bipartite Matching. We define a *vertex mapping* M between two vertex sets V and W as a set $M \subseteq V \times W$ that contains $\min(|V|, |W|)$ tuples and every vertex $v \in V \cup W$ is mapped to at most one other vertex, that is, v is contained in at most one tuple in M.

Given a matching cost for each pair of vertices $v \in V$ and $w \in W$ an *optimal vertex matching* is computed by solving the ASSIGNMENT PROBLEM: Given two sets A and B of the same size and a cost function c, the goal is to find a bijection π between A and B such that $\sum_{a \in A} c(a, \pi(a))$ is minimized. It can be computed in $\mathcal{O}(n^3)$ time where n is $\max(|V|, |W|)$ [AMO93, Theorem 12.2].

Note that if we want to match two vertex sets V, W and $|W| > |V|$, we add $|W| - |V|$ vertices V_0 to V and set the matching cost $c(v, w)$ to zero for $v \in V_0, w \in W$ to obtain a minimum cost vertex matching as an optimal solution to the assignment problem.

Dynamic Time Warping. A *warping path* $p_{n,m}$ between two time series A and B of lengths n and m is a list of L tuples $p_l := (i, j)$ that connect or warp data point of time series A at time step i to B's data point at time step j. It is guaranteed that the first and last time steps are mapped, so $p_1 = (1, 1)$, $p_L = (n, m)$ and time steps are only mapped going forward in time, i.e., $p_l \in \{(i_l, j_{l-1}), (i_{l-1}, j_l), (i_{l-1}, j_{l-1})\}$ for all $l \in [2, L]$. The cost of a warping path is the sum of differences between the data points on the path.

Dynamic Time Warping (dtw) measures the difference between two time series by a minimum cost warping path. Figure 2.1 illustrates how two time series are compared using dynamic time warping.

A minimum cost warping path is computable in $\mathcal{O}(n \cdot m)$ time using dynamic programming [SC78]. If we forbid to warp time steps that are more than w steps apart, that is, our warping path only contains tuples (i, j) such that $|i - j| \leq w$, then dynamic time warping becomes computable in linear time. We call the value w *warping window*. We define the relative window for two time series of length n and m as

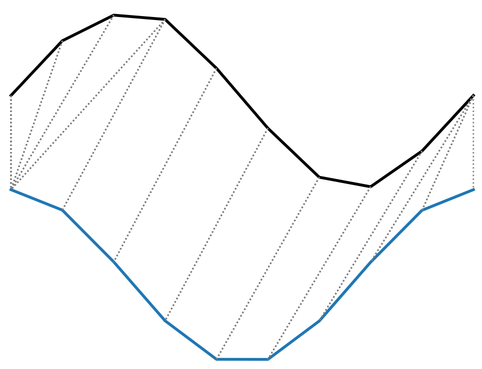
$$w_{\mathrm{rel}}(n, m) := w / \max(n, m).$$

Note that in every warping path between two time series of length n and m there are two time steps whose absolute difference is at least $|n - m|$. To measure the deviation from the diagonal of warping path $p_{n,m}$, we define

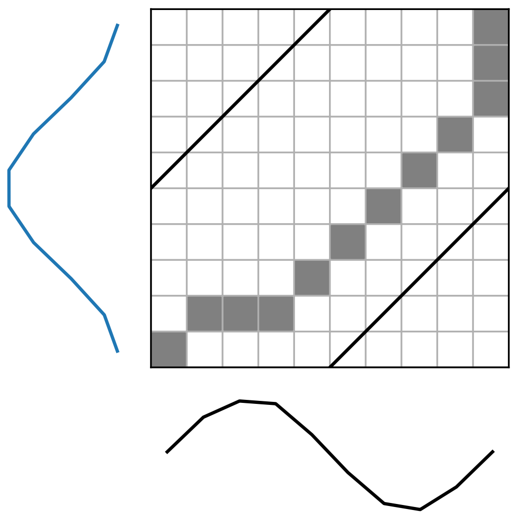
$$\mathrm{dev}(p_{n,m}) := \max_{(i,j) \in p_{n,m}} |i - j| - |n - m|.$$

The deviation relative to the length of two time series is the absolute deviation divided by their maximum length:

$$\mathrm{dev}_{\mathrm{rel}}(p_{n,m}) := \mathrm{dev}(p_{n,m}) / \max(n, m).$$



(a) Alignment of two time series.



(b) The mapping of the time series indices in a matrix, i.e., the warping path.

Figure 2.1: Illustration of time series comparison using dynamic time warping.

3 Methods for Comparing Temporal Graphs

We begin with concepts to measure the proximity of static graphs in Section 3.1. Based on them Section 3.2 presents methods to measure the (dis)similarity of temporal graphs.

3.1 Graph Proximity Measures

Two graphs G and H are *equivalent* if there exists an isomorphism between them, that is, a bijection $\pi : V(G) \to V(H)$ between their vertex sets $V(G)$ and $V(H)$ such that any two vertices $u, v \in V(G)$ are neighbors in G if and only if $\pi(u)$ and $\pi(v)$ are neighbors in H.

This motivates a proximity measure between two graphs based on a matching between their vertex sets as introduced by Jouili and Tabbone [JT09].

Graph Proximity using Vertex Matching. The *matching cost* C of a matching M between the vertex sets of two graphs G and H is the sum of all matched vertex pairs' distances. The distance is measured by a *distance function* $d : \mathbb{Q}^2 \to \mathbb{Q}$ based on two *vertex features* $f : V \to \mathbb{Q}$:

$$C(G, H, M) := \sum_{(v,w) \in M} d(f(v), f(w)).$$

The distance d might e.g. be the absolute difference of the vertex degrees. If two graphs do not have the same number of vertices, vertices $\overline{V}_M$ that are not matched are penalized by a cost $C_D(v)$:

$$C(G, H, M) := \sum_{(v,w) \in M} d(f(v), f(w)) + \sum_{v \in \overline{V}_M} C_D(v). \tag{3.1}$$

Let us denote all possible matchings between the vertex sets of two graphs G and H by $\mathcal{M}(G, H)$. Then the *vertex matching graph distance* dg is the cost of a cheapest matching $M \in \mathcal{M}(G, H)$:

$$\mathrm{dg}(G, H) := \min_{M \in \mathcal{M}(G,H)} C(G, H, M). \tag{3.2}$$

See also *assignment kernels* that are also based on the idea of vertex assignment [KGW16]. We briefly describe the ideas of other graph kernels and refer to the literature for more formal definitions. They are used afterwards for the temporal case.

Random Walk kernel. Random Walk Kernels measure the similarity of graphs by the number of common walks. Recall that two walks are common if they have the same color, that is, the same sequence of vertex (and edge) labels encountered on the walk. We follow the notation of Oettershagen et al. [Oet+b] and define a *k-step random walk kernel* by a feature map $\phi^k_{RW}(G)$ of graph G that counts the number of distinct colored walks in G up to length k. The kernel entry $\langle \phi^k_{RW}(G), \phi^k_{RW}(H) \rangle$ for two graphs G and H is the sum of products of the number of same colored walks up to length k.

Random Walk Approximation kernel. Instead of counting all distinct colored length-k walks, the approximation (or stochastic) variant of the walk kernel samples a fixed number S of walks at random from all possible walks in a graph G. An entry i of the feature map $\tilde{\phi}^k_{RW}(G)$ counts the number of temporal walks with color i normalized by S^{-1}. The kernel entry $\langle \tilde{\phi}^k_{RW}(G), \tilde{\phi}^k_{RW}(H) \rangle$ for two graphs G and H is the sum of products of the number of same colored length k-walks from S random sampled walks in G and H.

Weisfeiler-Lehmann subtree kernel. This kernel is based on the *color refinement algorithm* for graph isomorphism checking [She+11]. With each vertex having an initial label the algorithm computes in an iterative manner vertex labels based on the neighbors' labels for each vertex. The feature map $\phi^k_{WL}(G)$ of graph G counts the number of distinct labeled vertices in each iteration up to iteration k. Hence, kernel entry $\langle \phi^k_{WL}(G), \phi^k_{WL}(H) \rangle$ sums over the product of common labeled vertices in G and H according to the color refinement algorithm up to iteration k.

Graphlet kernel. The graphlet kernel due to Shervashidze et al. [She+09] is defined by a feature map that counts the occurrences of subgraphs of a fixed size, i.e., with k vertices where $k \in [3, 5]$ named *graphlets*. An entry of the kernel is the sum of products of the number of common graphlets.

In the following these principles are adapted to derive proximity measures for temporal graphs.

3.2 Temporal Graph Proximity Measures

First we introduce the comparison method by Froese et al. [Fro+20] based on a vertex matching. Then temporal graph kernels due to Oettershagen et al. [Oet+a; Oet+b] are presented.

3.2.1 Dynamic Temporal Graph Warping

Froese et al. [Fro+20] combine the ideas to compare static graphs and time series using *dynamic time warping*, see Section 2.3, to compare two temporal graphs by computing a mapping between the vertices and a warping path between the layers of the temporal graphs.

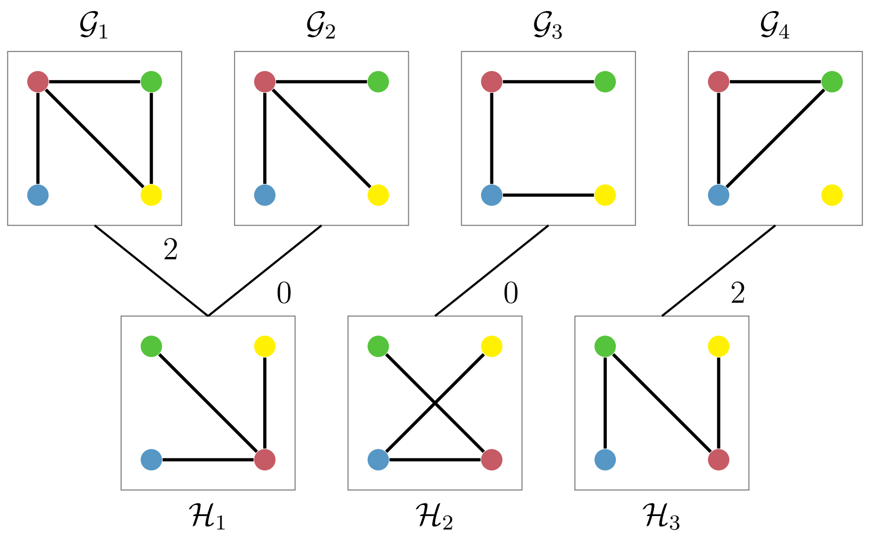


Figure 3.1: Example for dtgw-distance between temporal graphs $\mathcal{G}$ and $\mathcal{H}$. Vertices with the same color are matched. The connections between the temporal graphs' layers illustrate a warping path. Their labels are the distances between the layers. Herein, that is the sum of absolute differences of the matched vertex degrees. The resulting distance is 4.

The dissimilarity (or distance) is measured by the sum of the mapped vertex pairs costs' over all assigned pairs of layers in a warping path p. Let the set of warping paths between two temporal graphs $\mathcal{G}$ and $\mathcal{H}$ be $P(\mathcal{G}, \mathcal{H})$, the set of vertex mappings be $\mathcal{M}(\mathcal{G}, \mathcal{H})$, and the cost for a vertex mapping M between two layers $\mathcal{G}_i$ and $\mathcal{H}_j$ be $C(\mathcal{G}_i, \mathcal{H}_j, M)$, then the *dynamic temporal graph warping distance* is defined as

$$\operatorname{dtgw}(\mathcal{G}, \mathcal{H}) := \min_{M \in \mathcal{M}(\mathcal{G},\mathcal{H})} \min_{p \in P(\mathcal{G},\mathcal{H})} \sum_{(i,j) \in p} C(\mathcal{G}_i, \mathcal{H}_j, M). \tag{3.3}$$

An Example for the dtgw-distance between two temporal graphs is depicted in Figure 3.1.

On the negative side Froese et al. [Fro+20] show that this distance is NP-hard to compute, on the positive side they present a heuristic that performs well in practice.

DTGW Heuristic. The heuristic makes use of the observation that a warping path (or vertex matching) given a fixed vertex matching (or warping path) can efficiently be computed in polynomial time, see Section 2.3.

Initialized with either a warping path or a vertex matching the heuristic alternates between computing a vertex matching and warping path until the resulting distance value converges to a local minimum and does not improve. The running time is $\mathcal{O}(T^2 n + n^3 + n^2 T)$ for the maximum number of vertices n and the maximum lifetime T of two temporal graphs. Note that this computation is required for each pair of temporal graphs to compare.

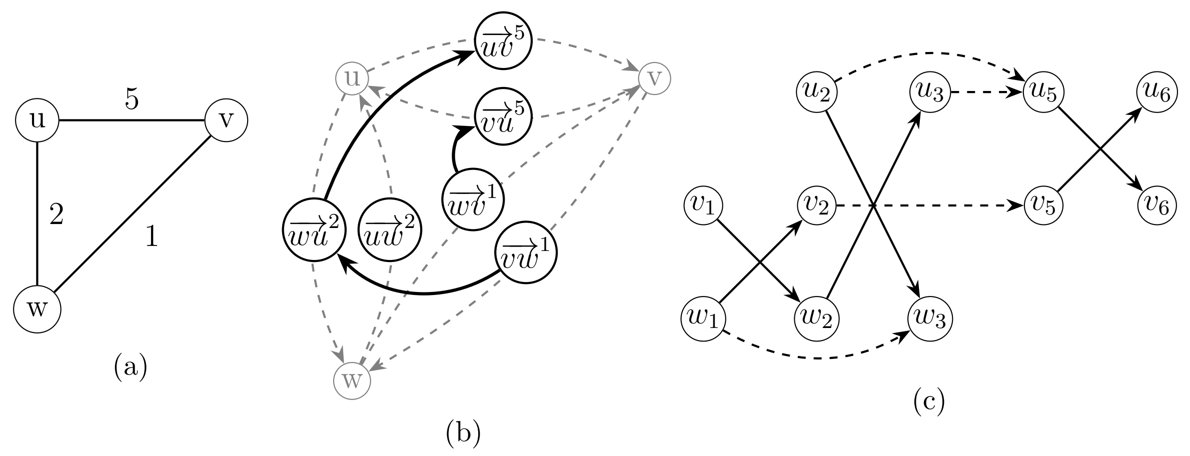


Figure 3.2: A temporal graph (a), its Directed Line Graph Expansion (b), and Static Expansion (c).

3.2.2 Temporal Graph Kernels

We briefly introduce two concepts for temporal graph kernels developed by Oettershagen et al. [Oet+a; Oet+b] and refer to their papers for more detailed and formal definitions. The first concept applies the idea of walk kernels, the second the idea of graphlet kernels for static graphs to the temporal case.

Static Expansion Kernels. Oettershagen et al. [Oet+b] map temporal graphs to "static graphs such that conventional static kernels can be applied, e.g., the random walk or Weisfeiler-Lehman subtree kernel. To catch temporal information, [they] use temporal walks". They use two directed static graphs that expand the temporal graph such that temporal walks are preserved.

Directed Line Graph Expansion. For each temporal edge $\{u, v\}$ in $\mathcal{G}_t$ two vertices $\overrightarrow{uv}^t$ and $\overrightarrow{vu}^t$ are inserted into the static graph. Then, each vertex $\overrightarrow{uv}^s$ is connected by a directed edge to all vertices $\overrightarrow{vx}^t$ if $t > s$. As an example see the Directed Line Graph Expansion in Figure 3.2b of the temporal graph shown in Figure 3.2a.

A vertex $\overrightarrow{uv}^t$ inserted for temporal edge $\{u, v\}$ in $\mathcal{G}_t$ is labeled $(l(u, t), l(v, t + 1))$ such that the vertex labels encountered on a walk in the temporal graph are the same as the labels of the corresponding walk in the directed line graph.

The resulting directed acyclic graph has size $\mathcal{O}(|\mathcal{E}|^2)$ for the set of temporal edges $\mathcal{E}$.

Static Graph Expansion. Static Graph Expansions play a role in various temporal graph problems [Mic15]. A Static Graph Expansion consists of a four vertices $u_t, v_t, u_{t+1}, v_{t+1}$ for each temporal edge $\{u, v\}$ in layer $\mathcal{G}_t$. The vertices are connected by directed edges going forward in time. To walk via temporal edge $\{u, v\}$ in $\mathcal{G}_t$ from u to v or from v to u edges (u_t, v_{t+1}) and (v_t, u_{t+1}) are inserted. To consider walks from u to w over two temporal edges $\{u, v\}$ in layer $\mathcal{G}_s$ and $\{v, w\}$ in $\mathcal{G}_t$ where $t > s+1$, a waiting edge (v_{s+1}, v_t) is inserted. At most one waiting edge per vertex to the smallest $t > s + 1$ is added. For an example Figure 3.2c shows the constructed Static Expansion Graph for the temporal graph in Figure 3.2a.

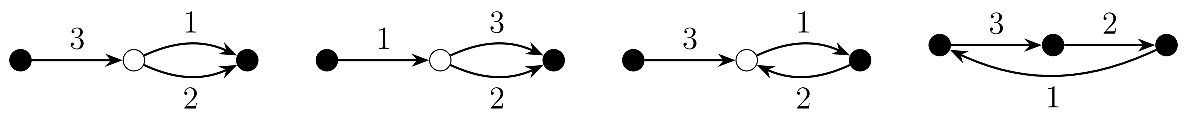


Figure 3.3: Temporal graphlets with 3 vertices. The edges' labels are the time steps at which they are present.

The size of the obtained directed acyclic graph is linear in the number of temporal edges $\mathcal{E}$.

Temporal Graphlet kernel. As the static graphlet kernel, see Section 3.1, the temporal graphlet kernel due to Oettershagen et al. [Oet+a] is defined by a feature map that counts temporal graphlets that were introduced by Paranjape, Benson, and Leskovec [PBL17] for directed temporal graphs. A *temporal graphlet* is a sequence of directed temporal edges $(u_1, v_1, t_1), \ldots, (u_n, v_n, t_n)$ that are chronologically ordered, $t_1 < \ldots < t_n$, and their induced static graph is connected. Two temporal graphlets $(v_1, u_1, t_1), \ldots, (v_n, u_n, t_n)$ and $(v'_1, u'_1, t'_1), \ldots, (v'_n, u'_n, t'_n)$ are *equivalent* if they have the same number of temporal edges, $n = n'$, the same number of vertices, and there exists a bijection π between the graphlets' vertices such that $\pi(u_i) = u'_i$ and $\pi(v_i) = v'_i$ for $i \in [n]$. Figure 3.3 shows four different temporal graphlets with three vertices. In a vertex labeled temporal graph we call the *color* of a graphlet its vertex label sequence $(l(v_1), l(u_1), \ldots, l(u_n))$. Two vertex labeled graphlets are equal if the graphlets without labels are equal and they have the same color. Oettershagen et al. [Oet+a] propose feature maps that count distinct colored temporal graphlets with two and three vertices and edges.

Note that computing a kernel requires $\mathcal{O}(N^2)$ computations of the inner product of two feature maps. Since that computation is efficiently possible, the running time is dominated by the computation of N temporal graphs' feature maps.

With these comparison methods at hand we move on to evaluate them experimentally on real world temporal graphs. The following chapter compares labeled temporal graphs that model dissemination processes.

4 Classify Dissemination Processes

In dissemination processes diseases, (fake) news, or computer viruses spread along temporal links in dynamic networks. They can be modeled as binary vertex labeled temporal graphs illustrated in Figure 1.1. Oettershagen et al. [Oet+b] present temporal graph kernels for classifying dissemination processes along side data sets of temporal graphs modeling these processes. We wish to find out whether the method of dynamic temporal graph warping due to Froese et al. [Fro+20] depicted in Section 3.2.1 can be a suitable proximity measure for that task.

Outline. Therefore, Section 4.1 adapts dynamic temporal graph warping (dtgw) to binary vertex labeled temporal graphs by designing vertex features and a distance function between temporal graph layers. Section 4.2 presents our experiments to classify dissemination processes. We begin with experiments regarding parameter sensitivity of the dtgw-heuristic before we present and discuss the results to classify dissemination processes.

4.1 Adapting Dynamic Temporal Graph Warping

The dtgw-distance defined in Equation (3.3) is based on a cost function that measures the dissimilarity between two temporal graphs' layers. That cost is the sum of assigned vertex pairs' distances given by a distance function between two vertex features, see Equation (3.1). Thus, to grasp the dissimilarity between vertex labeled temporal graphs, it is crucial to design expressive vertex features.

Vertex features. We propose two vertex features. The first captures the neighborhood of a vertex v up to a fixed distance k. That can be understood as features of trees rooted in v up to height k. The second is based on counting distinct colored walks from v up to some length k.

Subtrees. The most significant feature of a vertex in a labeled graph that models a dissemination process is its label. Thus, let feature $\sigma_t^0(v)$ of vertex v be label $l_t(v)$ at timestep t:

$$\sigma_t^0(v) := l_t(v).$$

Yet, only the label does not account for whether a vertex changes its label due to the labels of its neighbors. Therefore, the vertex neighborhood also wants to be considered.

The neighborhood is a set of labeled vertices that can be described by the number of vertices of each label. Let the binary labels be called "red" and "black" and the number

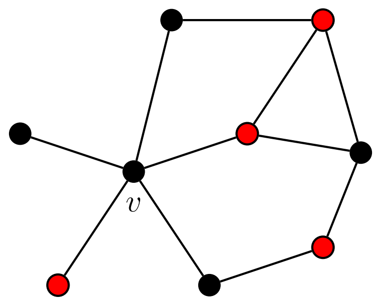


Figure 4.1: An illustration of the vertex features. The subtree vertex features of height $k \in [0, 2]$ are $\sigma^0(v) = 0$, $\sigma^1(v) = (0, \frac{2}{8}, \frac{3}{8})$, and $\sigma^2(v) = (0, \frac{2}{8}, \frac{3}{8}, \frac{2}{2\cdot 8}, \frac{1}{2\cdot 8})$. Let black vertices have label 0 and red ones label 1, then walk feature for length one is $\omega^1(v) = (3, 2, 0, 0)$ and for length two is $\omega^2(v) = (3, 8, 3, 1, 0, 0, 0, 0)$.

of red neighbors be denoted by $r_t(v)$ and the black ones by $b_t(v)$. Then the neighborhood of vertex v can be written as vector $\big(r_t(v), b_t(v)\big)$.

To account for the vertex label itself, vertex feature $\sigma_t^1(v)$ combines $l_t(v)$ with $r_t(v)$ and $b_t(v)$. Since the vertex label with value 0 or 1 is at least as important as its neighbors' labels, $r_t(v)$ and $b_t(v)$ are normalized to $[0, 1]$. To keep information about the neighborhood's size, the proportion of (infected) neighbors to the total number of vertices $|V|$ is taken:

$$\sigma_t^1(v) := \left(l_t(v), \tfrac{1}{|V|} r_t(v), \tfrac{1}{|V|} b_t(v)\right).$$

To refine that feature, let us explore the neighborhood further up to a depth k. Then feature vector $\sigma_t^k(v)$ concatenates the vertex label and the number of red and black labeled vertices of each neighborhood with a fixed distance up to distance k. Let

$$r_t^k(v) := \frac{1}{|V|} \sum_{u \in N_t^k(v)} l_t(u)$$

count the red labeled vertices in the k-neighborhood $N_t^k(v)$ normalized by $|V|^{-1}$ and $b_t^k(v)$ count the black labeled vertices, then

$$\sigma_t^k(v) := \big(l_t(v), r_t^1(v), b_t^1(v), \dots, r_t^k(v), b_t^k(v)\big).$$

The importance of neighborhood with distance k is considered by weighting r_t^k and b_t^k by an $\alpha_k \in [0, 1]$. Therefore the subtree vertex feature

$$\sigma_t^k(v) := \big(l_t(v), \alpha_1 r_t^1(v), \alpha_1 b_t^1(v), \dots, \alpha_k r_t^k(v), \alpha_k b_t^k(v)\big). \tag{4.1}$$

See Figure 4.1 for an example with $\alpha_k = k^{-1}$.

Walks. Similarly, consider the different colored walks beginning at a vertex. Recall that the color c of walk $w = (v_1, \dots, v_k)$ is the concatenation of vertex labels visited:

$$c_t(w) = \big(l_t(v_1), l_t(v_2), \dots, l_t(v_k)\big).$$

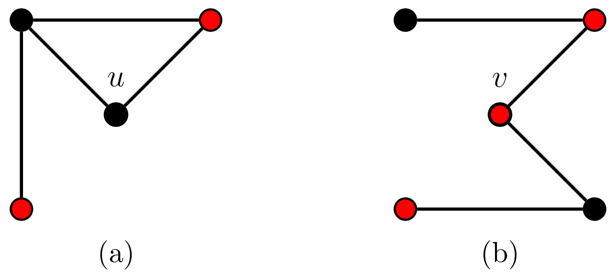


Figure 4.2: Example for distance between two vertex features. The subtree vertex features are $\sigma^2(u) = (0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{2\cdot 4})$ and $\sigma^2(v) = (1, \frac{1}{5}, \frac{1}{5}, \frac{1}{2\cdot 5}, \frac{1}{2\cdot 5})$. The L_1-distance is $||\sigma^2(u) - \sigma^2(v)||_1 = 1 + \frac{1}{20} + \frac{1}{20} + \frac{1}{10} + \frac{1}{40} = 1.225$.

Let $\mathcal{W}_t^l(v)$ be the walks of length l starting from vertex v at timestamp t, then the number of c-colored walks of length l is

$$\nu_t^l(v, c) := \left|\{w \in \mathcal{W}_t^l(v) \mid c_t(w) = c\}\right|.$$

Let vector $\varphi_t^l(v) \in [2^{l+1}]$ count the distinct colored walks from vertex v of length l such that each row of $\varphi_t^l(v)$ contains the number of length-l walks of a distinct color. Note that a walk color c with vertex labels 0 and 1 can be interpreted as a binary number $\text{BIN}(c)$. So formally row i of $\varphi_t^l(v)$ contains the number $\nu_t^l(v, \text{BIN}(i))$ of length-l walks starting from v in $\mathcal{G}_t$ of coloring $\text{BIN}(i)$. Thus, row $i \in [2^{l+1}]$ is defined as

$$\varphi_t^l(v)_i := \nu_t^l(v, \text{BIN}(i)).$$

Then, walk feature vector $\omega_t^k(v)$ counts the distinct colored walks from vertex v at timestamp t of length k:

$$\omega_t^k(v) := \left(\varphi_t^k(v)_1, \dots, \varphi_t^k(v)_{2^{k+1}}\right). \tag{4.2}$$

See Figure 4.1 for an instance as well.

Distances. We experimented with the L_1-distance, that is, the sum of absolute differences, also called Manhattan norm and the L_2-distance, that is, the Euclidean distance. Figure 4.2 gives an example for the distance between two vertex features.

Vertex deletion cost. If we compare temporal graphs with different numbers of vertices n and m, only $\min(n, m)$ vertices are matched. We consider a cost $C_D(v)$ for not matching vertex v. For "deletion" costs the L_1 and L_2 norm of vertex feature $f(v)$, zero, and the norm of $f(v)$ divided by the number of unmatched vertices $|n - m|$ for two temporal graphs with n and m vertices are used.

Note that a high cost may punish similar temporal graphs for having vertex sets of unequal size while a low cost may ignore vertices with high distance in distinct temporal graphs.

Table 4.1: Data set statistics. The number of temporal graphs per dataset is $|\mathcal{D}|$. The number of vertices of a temporal graph is denoted by $|V|$, the number of temporal edges by $|\mathcal{E}|$ and the lifetime by T.

Property	**Data set**					
	MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	DBPL	FACEBOOK
$\lvert\mathcal{D}\rvert$	97	180	200	373	755	995
$\max\lvert V\rvert$	20	60	50	99	60	100
$\varnothing\lvert V\rvert$	20	52.3	50	53.1	52.9	95.7
$\max\lvert\mathcal{E}\rvert$	3,363	589	505	190	275	181
$\varnothing\lvert\mathcal{E}\rvert$	734.6	272.4	229.9	99.9	160	134.5
$\max t$	5,577	204	49	90	47	105
$\varnothing t$	4,698	202.9	42.5	89.5	47	104.1

4.2 Experiments

We now want to experimentally evaluate how well the dtgw-heuristic can measure the proximity of temporal graphs that model dissemination processes to classify them.

First, data sets and classification tasks are introduced in Section 4.2.1. Preliminary experiments regarding the dtgw-heuristic's performance are done in Section 4.2.2. Then the experimental setup is described in Section 4.2.3 before the classification results are presented in Section 4.2.4.

4.2.1 Data Sets

Oettershagen et al. [Oet+b] created data sets of binary labeled temporal graphs that represent dissemination processes[1]. They simulated an infection spreading model on different publicly available real-world temporal graph data sets. These include face-to-face interaction networks of students, temporal social networks, and co-author graphs.

To obtain temporal graphs of different sizes, subgraphs were selected using breadth-first-search. The statistics of the resulting temporal graphs are listed in Table 4.1.

Dissemination Process Simulation. The dissemination process is simulated using a *susceptible-infected* (SI) model. After a random vertex is initially marked as infected, infections propagate along the temporal edges in the following time steps with a fixed probability: If vertex v has been newly infected at time step t, then it can infect a vertex w once in the next time step $t+1$ with a fixed probability p if w is a neighbor of v at time step $t+1$. Once a vertex has been infected it stays infected. The simulation terminates when half of the vertices are marked as infected.

Classification Tasks. Oettershagen et al. [Oet+b] run different dissemination simulations to obtain three classes of labeled temporal graphs and create random labeled graphs

[1] The data sets are available at `https://chrsmrrs.github.io/datasets`

in order to solve two classification tasks:

1. Identify dissemination processes: Distinguish random labeled graphs from graphs that model propagation processes. After an infection process is simulated on a temporal graph with propagation probability $p = 0.5$ another temporal graph is labeled randomly. A random subset of vertices with the same number of total infected vertices are labeled as infected at random time steps.

2. Classify different dissemination processes: Two classes of temporal graphs generated by dissemination simulations with different propagation probabilities $p = 0.2$ and $p = 0.8$ have to be differentiated.

Before we report the classification experiments that show how accurately the temporal graph proximity measures perform these tasks we experimentally evaluate how parameter choices of the dtgw heuristic affect its performance.

4.2.2 DTGW Heuristic Parameter Sensitivity

Recall that computing the dtgw-distance is an NP-hard minimization problem that is based on mapping two temporal graphs' vertices and temporal layers, see Section 3.2.1. To optimize the heuristic's performance in practice, we study the influence of three parameter choices for the heuristic.

To cancel out the impact of different vertex feature or distance metric choices on the analyzed correlations, subtree and walk vertex features with height $k \in [0, 2]$, length $k \in [2]$ and L_1-distance are computed and the results averaged.

First, we study the influence of the dtgw heuristic's initialization on the solution quality and running time.

Influence of heuristic initialization. The heuristic is initialized with either a mapping of the vertices or the temporal layers. Froese et al. [Fro+20] propose four heuristic initialization options:

1. *Shortest warping path* (swp). Choose a warping path of minimal length, that is, of length $\max(T_G, T_H)$ for liftetimes T_G, T_H.

2. *Optimistic warping path* (owp). Ignore a fixed vertex matching and find the cheapest warping path based on the minimal matching cost between two temporal layers i and j:
$$\mathrm{owp}(i, j) = \min_{M \in \mathcal{M}(G,H)} C(G_i, H_j, M).$$

3. *Product matching* σ^*. Find a minimal-cost vertex matching by considering the matching cost between two vertices u and v over all possible warpings between any two layers:
$$\sigma^*(u, v) = \sum_{i \in [T_G]} \sum_{j \in [T_H]} d(f_i(u), f_j(v)).$$

Table 4.2: Mean computation time of the dtgw-heuristic between two temporal graphs depending on the initialization in *ms*.

	Data set				
Init	MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	FACEBOOK
swp	345.7	40.6	18.2	22.4	54.7
owp	2,670	1,318.7	260.56	844.7	1,854
σ^*	603.5	192.1	52.94	133.9	285.6
σ_{opt}	573.2	190.1	29.37	135.1	293.1

4. *Optimistic matching* σ_{opt}. Find a minimal-cost vertex matching. The matching cost of vertices u and v is the minimal cost over all layers of the other temporal graph:

$$\sigma_{\text{opt}}(u, v) = \sum_{i \in [T_G]} \min_{j \in [T_H]} d(f_i(u), f_j(v)).$$

Computing a shortest warping path is done in $\mathcal{O}(T)$ time, while computing an optimistic warping path takes $\mathcal{O}(n^3 T^2)$ time and the initial vertex matching computations take $\mathcal{O}(n^2 T^2)$ time (since dynamic time warping takes $\mathcal{O}(T^2)$ time and minimum cost matching $\mathcal{O}(n^3)$ time, see Section 2.3). Note that based on the initialization the heuristic might converge to different local minima.

Table 4.2 reports the average running time for comparing two temporal graphs per initialization and data set. It can be observed that the initialization takes a considerable amount of computation time: The heuristic with an initial shortest warping path is 2 to 6 times faster than the vertex matching initializations, and up to 34 times faster than the heuristic with an initial optimistic warping path.

To see whether a high computation cost pays off, we measure the quality of a heuristic's solution depending on the initialization by the relative difference to a minimal solution found by any heuristic. The mean relative error between all temporal graphs' distances for each initialization and data set is reported in Table 4.3. Interestingly, the heuristic with an initial shortest warping path is also the best in terms of solution quality.

Therefore, in the following we initialize the dtgw-heuristic with a shortest warping path. Another important parameter to choose for the heuristic is analyzed in the next paragraph.

Influence of warping window. To compute the dtgw-distance, a minimum cost warping path has to be found. That is done via dynamic programming in $\mathcal{O}(T_G \cdot T_H)$ time for two time series of length T_G and T_H, see Section 3.2.1. It becomes computable in $\mathcal{O}(\max(T_G, T_H))$ time by restricting the warping path to a fixed window, also called Sakoe-Chiba-band cite. For an illustration see Figure 2.1.

This significant speedup of the heuristic's running time makes its computation for temporal graphs of long lifetimes tractable in practice e.g. for the MIT dataset, see Table 4.1. Also note that a warping path that deviates significantly from the diagonal can yield

Table 4.3: Relative error to the best solution found by any heuristic in %.

	Data set				
Init	MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	FACEBOOK
swp	0.04	0	1.44	0.16	0.44
owp	9.2	14.62	1.42	8.84	6.71
σ^*	8.27	14.05	2.87	9.52	7.87
σ_{opt}	8.7	14.59	3.74	11.01	8.56

Table 4.4: Relative deviation from the warping path's diagonal.

	Data set				
	MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	FACEBOOK
∅	0.29	0.31	0.06	0.18	0.21
max	0.93	0.83	0.45	0.54	0.67

smaller solution values, but might miss to capture the difference between dissemination processes over time. On the other hand, a narrow band might be too restrictive and worsen the result. Therefore, we study two questions:

1. How far does the warping path deviate from the diagonal?

2. How does the warping window's size influence the solution quality and running time?

Table 4.4 reports the mean and maximum deviation from the diagonal. Recall that by relative deviation from the diagonal warping path and by relative window size the absolute value is set relative to the maximum length of two time series, see Section 3.2.1. As the warping paths deviate on average 0.2 and up to 0.9 from the diagonal, we wish to analyze how a maximum warping window affects the solution quality and running time.

Figure 4.3 plots the mean error relative to a minimal solution found (by any warping window) and the mean running time for comparing two temporal graphs with respect to the relative warping window size. We see a quadratic decrease of the mean error and increase of the running time.

Since it is not clear if warping paths that deviate significantly from the diagonal to minimize distance values yield better classification results, we set in the following the relative warping window to 0.2. To keep the heuristic's computation for two temporal graphs of lifetimes n and m tractable in practice we limit the absolute warping window to $\min(0.2 \cdot \max(n, m), 100)$.

Influence of heuristic iterations. Recall that the dtgw heuristic computes a solution in an iterative manner by alternating between computing an optimal vertex mapping and warping path until it converges, see Section 3.2.1. This allows to stop the process after a maximum number of iterations.

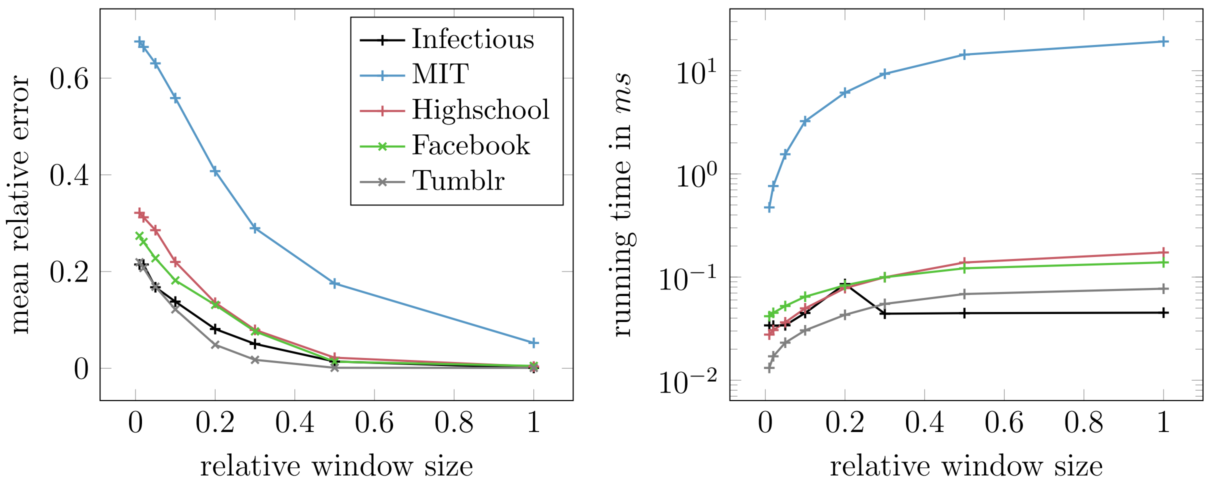


Figure 4.3: We plot the mean relative error and the running time w.r.t. the relative window size.

Table 4.5: Iterations until convergence.

	Data set				
	MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	FACEBOOK
min	3	2	2	2	2
∅	5.8	5.5	4.6	5.7	5.5
max	13	11	10	13	14

In the hope that the heuristic converges fast and thus allows to improve the running time while preserving the solution quality, we consider two questions:

1. How many iterations does the heuristic need to converge?

2. How does the number of iterations influence the solution quality and running time?

As reported in Table 4.5 the heuristic takes on average 5 and does not exceed 14 alterations to converge.

To see the effect of limiting the number of iterations on the heuristic's solution quality and running time we plot the mean relative error and average running time per comparison with respect to the maximum number of iterations in Figure 4.4. The mean relative error drops exponentially with the number of iterations. On the third iteration it is below 4%, improves up to the fifth iteration to a maximum of 1%, and converges slowly afterwards. Since the respective running time increases linearly with the number of iterations and the solution improves negligibly after the fifth iteration we set the maximum number of iterations to 5.

With that initial parameter choices for the dtgw-heuristic at hand we return to the experiments for classifying dissemination processes.

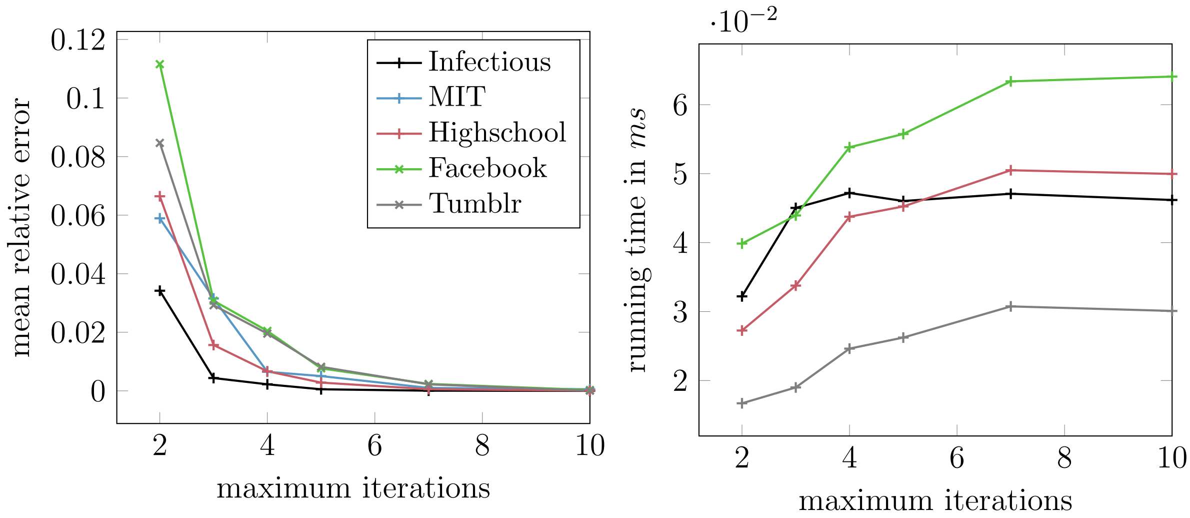


Figure 4.4: We plot the mean relative error and the running time w.r.t. the maximum number of iterations.

4.2.3 Experimental Setup

We report the methods used for classification based on the proximity measures of the concepts described above.

Classification methods. For classification Support Vector Machine (SVM) and Nearest Neighbor (NN) implementations of scikit-learn were used[2].

Remember that the dtgw-heuristic yields a distance between temporal graphs while the temporal graph kernels measure their similarity. While a SVM uses similarity scores for classification, NN classifier works with distance measures. To ensure that the classification results do not depend on the classification method distance matrices are transformed to kernels and vice versa and fed to both classifiers.

Distances and kernels. First, the computed distance matrices are normalized to $[0,1]$. We consider two kernels based on distance values: Radial-basis-function (rbf)-kernels

$$k_d^{\text{rbf}}(x,y) := \exp\bigl(-\gamma d(x,y)^2\bigr)$$

with hyperparameter γ and linear kernels

$$k_d^{\text{lin}}(x,y) := \langle x, x' \rangle_d^O = -\frac{1}{2}\bigl(d(x,y)^2 - d(x,O)^2 - d(y,O)^2\bigr),$$

where O is an arbitrary origin [HB04]. Kernels are transformed to distance matrices by their induced Euclidean distance:

$$d(x,y)^2 = \sum_i (x_i - y_i)^2 = \sum_i x_i^2 - 2x_iy_i + y_i^2 = k(x,x) - 2k(x,y) + k(y,y).$$

[2]Available at https://scikit-learn.org

Hyperparameters. The classification methods and proximity algorithms require the choice of hyperparameters. The slack C for SVM and the number of neighbors k for NN are chosen from $\{10^{-3}, 10^{-2}, \dots, 10^{3}\}$ and $\{1, 2, \dots, 10\}$ respectively. The rbf-kernel width γ is selected from $\{10^{-5}, 10^{-4}, \dots, 10^{1}\}$.

Initial parameter choices for the dtgw heuristic are given in Section 4.2.2. The subtree and walk vertex features are computed for height $k \in [0, 3]$, walk length $k \in [3]$, with weights $\alpha_k = 0.5^k$ and $\alpha_k = k^{-1}$. Their distances are measured by the L_1 and L_2 norm. For the vertex deletion cost $C_D(v)$ the values 0, the norm of vertex feature $f(v)$, and the norm of $f(v)$ divided by the number of unmatched vertices were considered.

The Weisfeiler-Lehmann subtree kernel and the random walk kernel were computed for depth $k \in [3]$. The temporal graphlet kernels that count the number of labeled triangles graphlets, star graphlets, and both were computed.

Model Selection. The classification accuracy of all parameter choices for each proximity method was computed. Thereby, the classification method and its best hyperparameters were chosen by 5-fold cross-validation. Each cross validation was repeated 5 times with different random folds to obtain the average classification accuracy and standard deviation for each model.

Implementation. To make the computation of dtgw-distance matrices for N temporal graphs that require $\mathcal{O}(N^2)$ pairwise dtgw-heuristic computations better tractable in practice the heuristic has been re-implemented in C and parallelized. The code features a python interface and is made publicly available including all scripts for the experimental evaluation[3].

All experiments were conducted on a workstation with an Intel Xeon W-2125 with 8 cores @4.0GHz and 27GB of RAM running Ubuntu 18.04.6 LTS. It was compiled using GNU C Compiler 7.5.0.

To limit the computation time of the dtgw heuristic a subset of 100 instances per data set was selected.

4.2.4 Results

We report the best classification accuracy for each algorithm and data set for the first classification task in Table 4.6 and for the second task in Table 4.7. Recall that the first task is to identify dissemination processes and the second task is to distinguish two different dissemination processes.

Temporal graph kernels and dtgw achieve similar classification accuracy on the identification task except for two datasets – Tumblr and Facebook. However, the different dissemination processes are most accurately classified on all data sets by the temporal graphlet kernel (except for Highschool by a half percent). Dtgw performs slightly worse than temporal walk kernels on this task.

The subtree vertex feature perform better than the walks feature. The results for all parameter choices for each method are appended in Chapter 7. Note that taking

[3]https://github.com/jonasschultemattler/cmptgraphs

Table 4.6: Classification accuracy in % for the identification task.

Algorithm		Data set					
		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	**95.9**$\pm_{5.0}$	92.0$\pm_{2.4}$	96.0$\pm_{3.7}$	63.0$\pm_{12.9}$	**99.0**$\pm_{2.0}$	80.0$\pm_{8.4}$
	walks	73.2$\pm_{12.5}$	83.0$\pm_{8.7}$	87.0$\pm_{6.8}$	63.0$\pm_{8.1}$	84.0$\pm_{8.0}$	77.0$\pm_{5.1}$
tkernel	SE	92.8$\pm_{4.5}$	**95.8**$\pm_{4.2}$	94.2$\pm_{4.8}$	89.6$\pm_{6.3}$	98.0$\pm_{2.6}$	94.0$\pm_{6.5}$
	LG	91.5$\pm_{5.6}$	93.2$\pm_{6.4}$	94.8$\pm_{6.3}$	**92.0**$\pm_{5.1}$	**99.0**$\pm_{2.0}$	**95.0**$\pm_{5.5}$
	TGK	91.3$\pm_{7.2}$	94.0$\pm_{2.0}$	**97.0**$\pm_{2.4}$	90.0$\pm_{7.1}$	98.0$\pm_{2.4}$	93.0$\pm_{4.0}$

Table 4.7: Classification accuracy in % for the differentiation task.

Algorithm		Data set					
		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	66.1$\pm_{4.7}$	87.0$\pm_{4.0}$	80.0$\pm_{5.5}$	68.0$\pm_{2.4}$	72.0$\pm_{4.0}$	79.0$\pm_{2.0}$
	walks	55.7$\pm_{7.6}$	69.0$\pm_{13.6}$	64.0$\pm_{8.0}$	61.0$\pm_{4.9}$	64.0$\pm_{10.2}$	69.0$\pm_{11.1}$
tkernel	SE	67.3$\pm_{9.1}$	**91.6**$\pm_{5.4}$	86.4$\pm_{10.8}$	75.0$\pm_{12.6}$	83.4$\pm_{7.8}$	81.8$\pm_{7.0}$
	LG	**84.2**$\pm_{7.5}$	90.6$\pm_{7.1}$	**87.4**$\pm_{9.1}$	76.0$\pm_{11.4}$	81.8$\pm_{9.4}$	81.2$\pm_{8.0}$
	TGK	81.9$\pm_{8.8}$	91.2$\pm_{7.7}$	87.0$\pm_{10.0}$	**81.0**$\pm_{8.6}$	**90.2**$\pm_{7.8}$	**82.2**$\pm_{8.7}$

only the vertex label (subtree vertex feature with $k = 0$) into account does the same as comparing the number of infections over time using dynamic time warping. It already achieves a high classification accuracy to distinguish dissemination processes from random infections, whereas subtree vertex features that capture the neighborhood ($k > 0$) could differentiate dissemination processes better.

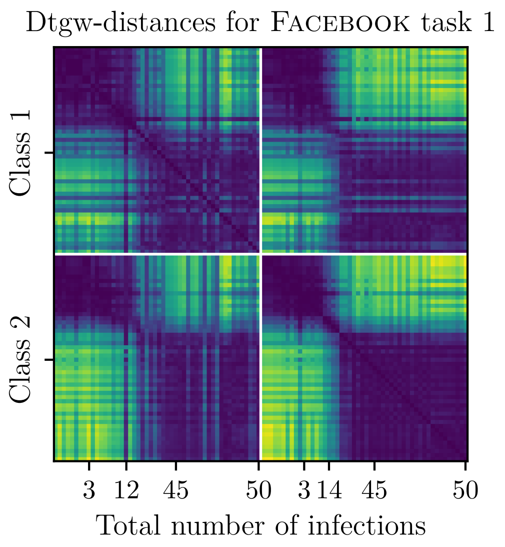
Also note that dtgw achieves high classification accuracy on the MIT and Infectious data sets that only contain temporal graphs with equal number of vertices. A deletion cost for unmatched vertices improved the classification accuracy on the Tumblr and DBPL data set for the subtree vertex feature, on the Highschool data set zero was better. The distance norm did not have an observable influence on the classification accuracy.

For the sake of completeness, we mention that Nearest Neighbor was the best classifier for distances, and linear SVM for kernels. The RBF-kernel did not improve the classification given kernels or distances.

The running times for the methods that achieved the best classification accuracy are appended in Table 7.3 and Table 7.8. The running time of the dtgw-heuristic is by up to three orders of magnitude higher than the temporal graph kernels. Recall that the heuristic has to be computed for each pair of temporal graphs. On the MIT data set containing temporal graphs whose lifetime is on average almost five thousand it takes 45 minutes to compute all distance pairs for 100 temporal graphs with the dtgw-heuristc.

Explanation. We try to give explanations for the classification results.

Reason 1. Temporal graphs with high difference in the number of overall infected



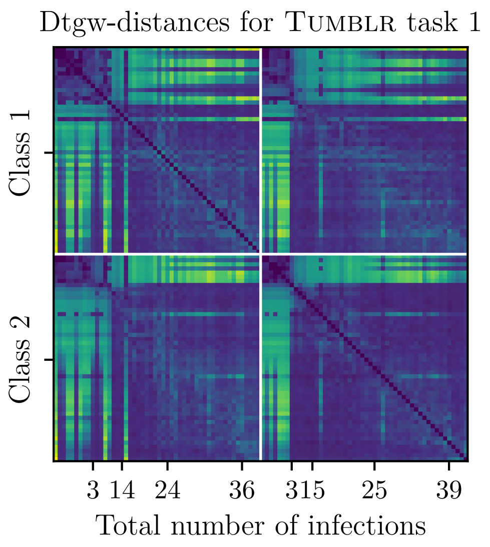


Figure 4.5: Dtgw-distances between all temporal graphs of the two data sets with worst classification accuracies for the identification task. The temporal graphs are grouped by the class they belong to and within each class sorted by the total number of infected vertices.

vertices have a high dtgw-distance independent of the kind of dissemination process.

A look at the Tumblr and Facebook data sets of the first identification task on which the dtgw-heuristic performed bad shows that they contain 9 instances with at most 3 total infections in the dissemination process. Figure 4.5 shows the dtgw-distances for the data sets grouped by class and within each class sorted by the total number of infected vertices. Observe that the distances between temporal graphs with few infected vertices and all other temporal graphs are high. As dtgw measures two temporal graphs' vertex differences over time the distance between propagation processes with a big difference in the number of infected vertices is high – no matter which class they belong to. On the other hand a kernel that measures similar temporal graphs' patterns might grasp small similarities of propagation processes.

To find out whether dtgw is capable to distinguish dissemination processes in which the same amount of vertices have been infected in the end, we filtered the data sets by temporal graphs whose proportion of infected vertices is 50% and repeated the experiment. That improved the classification accuracy for the identification task on the Facebook and Tumblr data sets for dtgw-distance and the temporal graph kernels (the results are appended in Table 7.5 and Table 7.10). Dtgw remained the worse proximity measure.

Reason 2. Matched vertex pairs that do not differentiate dissemination processes increase the dtgw-distance.

Consider two temporal graphs $\mathcal{G}$ and $\mathcal{H}$ shown in Figure 4.6a. In $\mathcal{H}$ an infected vertex infects a neighbor along a temporal edge and both vertices are connected in the time steps to follow, in $\mathcal{G}$ they are not. If the the neighborhood is part of the vertex feature, the dtgw-distance can get arbitrary large though $\mathcal{G}$ and $\mathcal{H}$ model the same dissemination

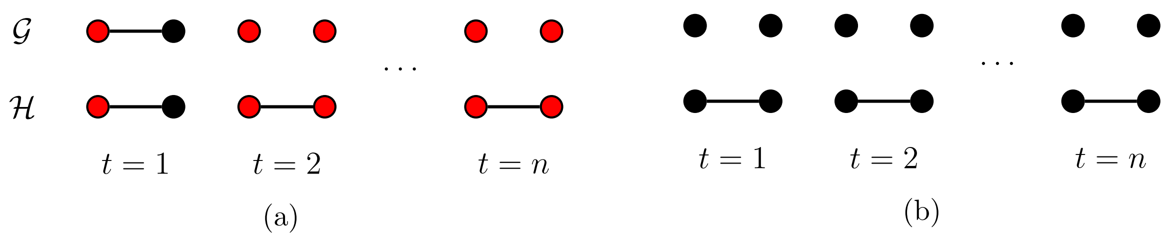


Figure 4.6: Two pairs of temporal graphs modeling a similar dissemination process whose dtgw-distance can be high.

process. To address this, we adapt the vertex subtree feature. We only count infected neighbors if the vertex itself is not infected, i.e., its label $l_t = 0$:

$$\tilde{r}_t^k(v) := \begin{cases} r_t^k(v) & \text{if } l_t(v) = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Recall that $l_t(v) \in \{0, 1\}$ is the label of vertex v at time step t, $b_t^k(v)$ and $r_t^k(v)$ are the numbers of (not) infected vertices in the neighborhood of v with distance k at t.

Analogously, vertex pairs that are not infected can increase the dtgw-distance by having different numbers of non-infected neighbors. Figure 4.6b illustrates a simple case. Since they do not differentiate dissemination processes, we only count not infected neighbors after a vertex has been infected:

$$\tilde{b}_t^k(v) := \begin{cases} b_t^k(v) & \text{if } l_t(v) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Then the subtree vertex feature defined in Equation (4.1) is adapted to

$$\tilde{\sigma}_t^k(v) := \big(l_t(v), \alpha_1 \tilde{r}_t^1(v), \alpha_1 \tilde{b}_t^1(v), \ldots, \alpha_k \tilde{r}_t^k(v), \alpha_k \tilde{b}_t^k(v)\big). \tag{4.3}$$

Table 4.8 and Table 4.9 report the best classification accuracy on the filtered data sets for the old subtree vertex feature σ and the adapted feature $\tilde{\sigma}$. The dtgw-heuristic with the adapted feature achieves slightly better classification results, noticeably for the differentiation task. Still, it remains a worse proximity measure than the temporal graph kernels on the Tumblr and Facebook data set for the identification task and the differentiation task on all data sets but the Infectious data set.

Finally, note that dtgw achieves high accuracies on the MIT and Infectious data sets that only contain temporal graphs with equal number of vertices. If there is a difference in the number of vertices participating in a dissemination process a vertex matching between most common vertices might not differentiate the processes reasonably.

4.3 Discussion

We focused on the adaption of the dtgw algorithm due to Froese et al. [Fro+20] to binary labeled temporal graphs in order to compare its performance to classify dissemination

Table 4.8: Classification accuracy in % for the identification task on data sets that contain temporal graphs whose proportion of infected vertices is 50%.

Algorithm		Data set					
		MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	DBPL	FACEBOOK
dtgw	subtrees σ	95.9±5.0	90.0±3.2	97.0±2.4	71.0±6.6	**100.0**±0.0	84.0±3.7
	subtrees $\tilde{\sigma}$	**97.9**±4.2	93.0±5.1	97.0±4.0	70.0±3.2	**100.0**±0.0	85.0±5.5
tkernel	SE	92.8±4.5	**98.6**±2.2	97.0±4.3	**100.0**±0.0	98.2±2.7	**100.0**±0.0
	LG	91.5±5.6	97.8±2.7	**97.6**±2.5	**100.0**±0.0	**100.0**±0.0	**100.0**±0.0
	TGK	91.3±7.2	97.0±2.8	97.4±3.8	**100.0**±0.0	99.0±2.0	**100.0**±0.0

Table 4.9: Classification accuracy in % for the differentiation task on data sets that contain temporal graphs whose proportion of infected vertices is 50%.

Algorithm		Data set					
		MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	DBPL	FACEBOOK
dtgw	subtrees σ	66.1±4.7	87.0±10.3	80.0±7.1	72.0±9.3	62.0±7.5	70.0±8.9
	subtrees $\tilde{\sigma}$	73.0±6.6	89.0±3.7	**84.0**±8.6	75.0±10.5	63.0±6.8	70.0±7.1
tkernel	SE	67.3±9.1	89.2±5.9	83.2±8.5	82.2±8.1	85.0±8.2	**83.6**±7.6
	LG	63.0±8.4	90.6±7.1	83.6±8.6	77.2±9.4	85.8±6.7	79.0±9.7
	TGK	**81.9**±8.8	**91.4**±6.2	80.0±3.2	**82.6**±8.4	**92.6**±5.7	79.0±10.0

processes to the recently proposed temporal graph kernels of Oettershagen et al. [Oet+a; Oet+b] on the data sets published by them.

We found that temporal graph kernels are mostly a better proximity measure for temporal graphs that model dissemination processes than the dtgw-distance. The difference between most common vertices during the process might not differentiate dissemination processes as well as common patterns (graphlets) in the temporal graphs that capture the moments of dissemination. Moreover, if there is a considerable difference in the number of vertices that are part of a dissemination process or in the total number of vertices, a vertex matching is not reasonable and dtgw does not perform well.

Nevertheless, dtgw offers a flexible method that can be adapted to different applications. Therein, it allows to include expert knowledge to design vertex features. Solving a minimization problem for each pair of temporal graphs to compare, even with a tuned heuristic, may be intractable in practice for big numbers of temporal graphs. Yet, it offers an explanation why two temporal graphs are similar or behave differently by looking at the vertex mapping and the time warping path. This can have applications in de-anonymization temporal networks [Fro+20]. Computing a vertex mapping and warping path for each pair of temporal networks might pay off if the reason for similarity or the role of vertices is interesting. In the case of dissemination that could be identifying individuals that either tend to contribute to the propagation process or block it.

By adapting dynamic temporal graph warping to the application of dissemination processes, contrasting its (dis-)advantages, and by providing an efficient implementation of the dtgw-heuristic we hope to contribute this method to the repertoire of temporal graph comparison methods for other applications.

5 Connectome Classification using Temporal Graphs

In this chapter we wish to propose a novel approach to compare human brain activity by modeling the brain's dynamic functional connectivity as a temporal graph.

Because "the connectivity and functional activity between neurons in the human brain are indicative of diseases such as Alzheimer's disease as well as subjects' reactions to sensory stimuli [...], researchers in neuroscience have studied the similarities of brain networks among human subjects" [KJM20b]. Recently, functional Magnetic Resonance Imaging (fMRI) allowed to measure the dynamic activity of human brains that aided work "to understand the dynamic interplay of the brain's networks. The intent of this research is that it will yield insight into the complex and dynamic nature of cognitive human abilities" [TBF17].

We aim to contribute to that research by modelling dynamic brain activities captured by fMRI as temporal graphs. We develop similarity measures for these temporal brain networks and apply them and other temporal graph proximity measures to distinguish between adults and children, patients with and without Autism and ADHD-disorder. We further compare different brain regions' behaviour to find patterns that correlate them.

Outline. Section 5.1 surveys related work that use fMRI data to study brain activity. In Section 5.2 our approach to build temporal graphs given brain activity (fMRI) signals is described. Then, Section 5.3 introduces proximity measures for the brain activity signals and in particular temporal brain networks. Section 5.4 presents our experiments to classify brain diseases using temporal graphs. In Section 5.5 dynamic temporal graph warping is used to detect similar brain regions. The results and future work are discussed in Section 5.6.

5.1 Related Work

Farahani, Karwowski, and Lighthall [FKL19] survey work on brains' functional connectivity modeled as static graphs. As fMRI makes it possible to capture the brain's temporal evolving activity its dynamic functional connectivity can be analyzed. See [Cal+14; DFC18; Hut+13; Lur+20] for surveys on dFC in resting fMRI. Recently Graph Neural Networks were used to model the brain's dFC [KYK21; Wei23].

To the best of our knowledge only Thompson, Brantefors, and Fransson [TBF17] model brain connectivity as a temporal graph. They use temporal vertex centrality measures to investigate which brain regions play a central role using fMRI data acquired of patients

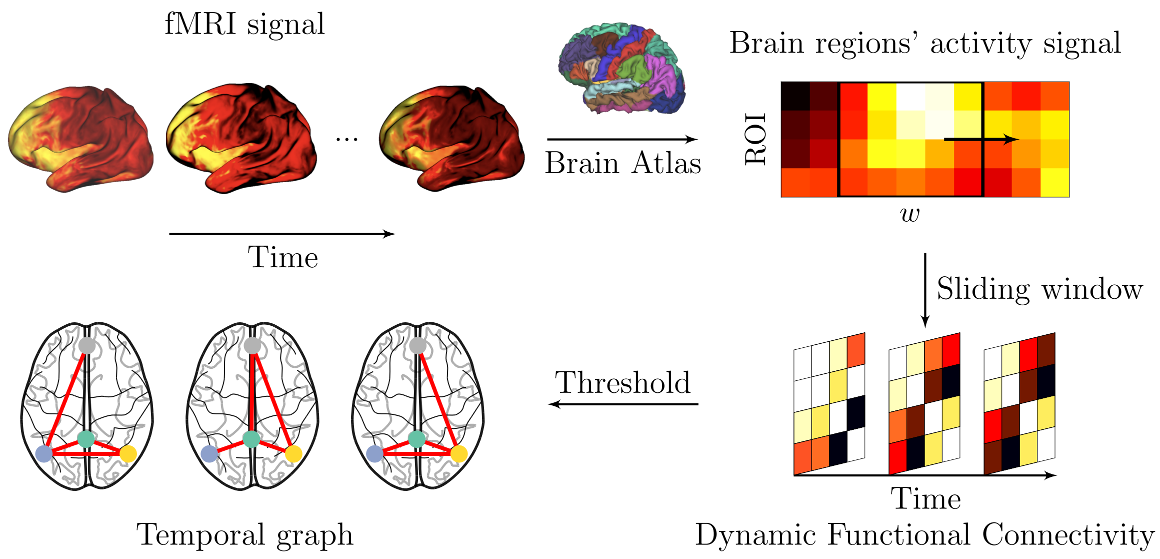


Figure 5.1: Procedure to construct a temporal graph from a fMRI signal. Given fMRI data and a brain atlas the brain regions' activity over time is calculated in step 1. Using a sliding window approach the brain regions' dynamic functional connectivity is estimated – step 2. Finally, in step 3 a threshold determines a functional connection that corresponds to a temporal edge between brain regions in the constructed temporal brain network.

during open and closed eyes condition. We follow their approach to model the dynamic functional brain connectome as a temporal graph described in the following Section 5.2 but use a different technique to estimate brain regions' functional connectivity.

5.2 Brain Connectivity as a Temporal Graph

Through functional Magnetic Resonance Imaging (fMRI) the brain's functional activity can be measured. In particular, blood-oxygen-level-dependent (BOLD) activity is measured using MRI that indicates functional activity. Given MRI measurements over a period of time, the brain's dynamic functional connectivity is naturally modeled by a temporal graph. The brain's activity is organized into networks that divide the brain into functional *regions of interest* (ROIs). These regions are the set of vertices. If there is a *functional connectivity* between two regions v and w at time step t, there exists an edge $\{v, w\}$ in layer $\mathcal{G}_t$.

In the following we describe our methodology to extract functional connectivity from fMRI data and build a temporal graph that models a brain's dynamic functional connectivity. The procedure is illustrated in Figure 5.1.

1. Extract Brain Regions Activity. Given a three dimensional atlas of the brain that defines N functional segregated brain regions of interest a region's activity $A(t)$ at time step t is calculated by the mean BOLD signal at t in that region. For a fMRI signal of length T this results in a brain region activity signal $A \in N \times T$.

2. Estimate Functional Connectivity. Yet, it is not obvious how the regions' activities correspond to functional connections. To estimate functional connectivity between two regions at time step t, not only the activity signal at t has te be taken into account [TF18]. We follow a popular approach to compute the regions' correlation within a time window around t [DFC18; KYK21].

A time window of length w is used to extract activity matrices $\tilde{A}(t) \in N \times w$ that contain the activity signal A from time step t to $t + w$. The window is shifted by Δt time steps to obtain $\lfloor (T - w)/\Delta t \rfloor$ windowed activity matrices. Then, the *functional connectivity* $FC_{ij}(t)$ between region i and j at time step t is defined by the regions' correlation in $\tilde{A}(t)$:

$$FC_{ij}(t) := \frac{\operatorname{cov}(\tilde{A}_i(t), \tilde{A}_j(t))}{\sigma_{\tilde{A}_i(t)} \sigma_{\tilde{A}_j(t)}}$$

(cov is the covariance and $\sigma_{\tilde{A}_i(t)}$ is the variance of $\tilde{A}_i(t)$).

3. Build Temporal Graphs. If the correlation of two regions is above a threshold, a functional connection is assumed and an edge between the respective vertices is inserted into the temporal graph.

The parameter choices to build the temporal graphs are reported in Section 5.4.2. To measure the proximity of the built temporal graphs we used the following methods.

5.3 Algorithms

To see whether the brain regions' activity signal itself is useful for classification they are compared for a baseline. We move on towards classification using functional connectivity by looking at the brain regions' connectivity over the whole period of time. Then, proximity measures for temporal graphs that model dynamic brain connectivity are presented. Note that these temporal graphs are not binary vertex labeled temporal graphs as in Chapter 4.

Comparing Brain Regions' Activity Signal. The brain regions' activity signals are compared using Dynamic Time Warping, see Section 2.3. The L_1- and L_2-distance between two activity signals $A(i)$ and $A(j)$ at time steps i and j are used as costs to warp time step i to j. Nearest Neighbor is used for classification.

Comparing Brain Regions' Correlation. The correlation of brain regions activity is computed throughout the whole signal to extract functional connectivity between the brain regions over the entire period of time. The obtained connectivity matrices are flattened and then used as feature vectors for comparison with a SVM.

This serves as a baseline for classification using brain regions functional connectivity. Dynamic functional connectivity is considered by comparing the temporal brain networks.

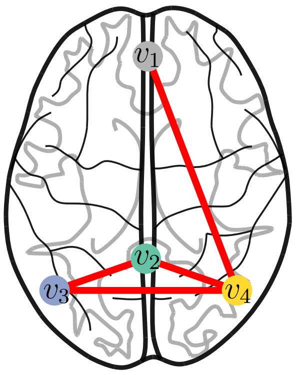


Figure 5.2: An illustration of the vertex features. The Interaction features of vertex v_2 that indicate a connection to vertices $v_1, \ldots, v_4$ for height $k \in [2]$ and weight $\alpha_k = k^{-1}$ are $I^1(v_2) = (0, 0, 1, 1)$ and $I^2(v_2) = (0, 0, 1, 1, \frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2})$. The degree features for an $\alpha_k = (\frac{1}{2})^k$ are $\delta^1(v_2) = (2, \frac{1}{2}, 0, \frac{2}{2}, \frac{2}{2})$ and $\delta^2(v_2) = (2, \frac{1}{2}, 0, \frac{2}{2}, \frac{2}{2}, \frac{1}{4}, 0, \frac{2}{4}, \frac{2}{4})$.

Comparing Temporal Brain Networks

First we propose the method temporal graph warping. After that temporal graph kernels are adapted to compare not binary labeled temporal graphs.

Temporal Graph Warping

When comparing temporal brain networks observe that a matching between the vertices that represent the brain regions already exists. Given a vertex mapping the difference between two brains' functional connectivity over time can be measured using dynamic time warping, see Section 2.3.

Therefore, a proximity measure between the functional connectivity of two time steps is needed. We sum over the distances between the brain regions that are given by a distance metric between two feature vectors that reflect the brain region's connections. Hence, vertex features that accurately represent a brain region's connectivity are required.

Vertex features. We propose two vertex features.

Interaction feature. If there is a functional connectivity between a brain region u and v at time step t, then v is a neighbor of u at time step t, $v \in N_t(u)$. Let

$$\nu_t(u, v) := \begin{cases} 1 & \text{if } v \in N_t(u), \\ 0 & \text{otherwise} \end{cases}$$

indicate whether there is a functional connectivity between two brain regions u and v at time step t, then the regions or vertices $v_1, \ldots, v_n$ a brain region u interacts with at time step t can be expressed by a binary vector

$$I_t(u) := \left(\nu_t(u, v_1), \nu_t(u, v_2), \ldots, \nu_t(u, v_n)\right).$$

To refine that feature take the interacting neighbors of the neighbors into account. Let k be a natural number and

$$\nu_t^k(u,v) := \begin{cases} 1 & v \in N_t^k(u), \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$I_t^k(u) := \left(\nu_t^1(u,v_1), \ldots, \nu_t^1(u,v_n), \nu_t^2(u,v_1), \ldots, \nu_t^k(u,v_n)\right)$$

captures the interacting neighbors up to the k-th neighborhood $N_t^k(u)$. The influence of neighborhood with distance k on the vertex feature is included with weight $\alpha_k \in [0,1]$:

$$I_t^k(u) := \left(\alpha_1\nu_t^1(u,v_1), \ldots, \alpha_1\nu_t^1(u,v_n), \alpha_2\nu_t^2(u,v_1), \ldots, \alpha_k\nu_t^k(u,v_n)\right). \tag{5.1}$$

Weights $\alpha_k = 0.5^k$ and $\alpha_k = k^{-1}$ were considered. See Figure 5.2 for an example.

Observe that the L_1-distance between two feature vectors of vertices u and v is equivalent to $\sum_k (|N_k(u)| + |N_k(v)| - |N_k(u) \cap N_k(v)|)\alpha_k$.

Degree feature. Similarly, a brain region's connectivity can be measured by the number of interacting regions (the vertex degree) and further by the degree of its neighbors and their neighbors. The resulting feature vector $\delta_t^k(u)$ of vertex u at timestep t that captures the vertex degrees of itself and its neighbors up to the k-th neighborhood is definded as:

$$\delta_t^k(u) := \left(\delta_t(u), \alpha_1\mu_t^1(u,v_1), \ldots, \alpha_1\mu_t^1(u,v_n), \alpha_2\mu_t^2(u,v_1), \ldots, \alpha_k\mu_t^k(u,v_n)\right), \tag{5.2}$$

with

$$\mu_t^k(u,v) := \begin{cases} \delta_t(v) & v \in N_t^k(u), \\ 0 & \text{otherwise} \end{cases}$$

indicating the degree $\delta_t(v)$ of a neighboring vertex v of u at timestep t with distance k and α_k the weight for a neighbor's degree with distance k. For instance see Figure 5.2.

Temporal Graph Kernels

We adapt the temporal expansion graph kernels [Oet+b] for binary vertex labeled graphs, see Section 3.2.2, to temporal graphs without vertex labels.

Therefore, each vertex (brain region) is assigned a distinct label. The resulting feature maps of the Random Walk and Weisfeiler-Lehmann kernels for walk length and height k have size $\mathcal{O}(|V|^k)$. That requires k to be small such that the kernels can be computed efficiently. The random walk approximation kernel, see Section 3.1, allows for bigger k by sampling a constant number S of length-k random walks.

Also recall that the Directed Line Graph Expansion has a space complexity of $\mathcal{O}(|\mathcal{E}|^2)$ for the number of temporal edges $|\mathcal{E}|$.

Table 5.1: Data set statistics. The number of temporal graphs per dataset is $|\mathcal{D}|$. The number of temporal edges is denoted by $|\mathcal{E}|$ and the lifetime by T.

Property	**Data set**		
	AUTISM	DEVELOPMENT	ADHD
$\lvert\mathcal{D}\rvert$	100	100	40
$\max\lvert\mathcal{E}\rvert$	26,635	20,727	36,786
$\varnothing\lvert\mathcal{E}\rvert$	21,132.2	20,723.4	22,174.8
$\max T$	52	40	71
$\varnothing T$	41.3	40	42.8

5.4 Experiments

First, we depict the fMRI data sets and classification tasks. Afterwards the parameters for temporal graph construction and the experimental protocol are reported before the results are presented.

5.4.1 FMRI Data sets and Classification Tasks

Three publicly available fMRI data sets were used:

Autism data set. Nielsen et al. [Nie+13] collected fMRI data of 964 patients with and without Autism Spectrum Disorder, preprocessed, and published them. See [San+22] for a survey of Autism Spectrum Disorder (ASD) classification using fMRI.

Brain development data set. Richardson et al. [Ric+18] studied differences between brains of 122 children (3–12 years) and 33 adults who watched a short movie while undergoing fMRI and made the data available.

ADHD data set. FMRI data of 973 individuals with and without Attention Deficit Hyperactivity Disorder (ADHD) was preprocessed[1] and published [Bel+17].

All data sets were downloaded using the python package `nilearn`[2]. Due to the size of fMRI data a subset of at most 100 subjects was randomly selected per data set.

We try to distinguish children and adults on the data set DEVELOPMENT and patients with and without Autism and ADHD on the AUTISM and ADHD data set respectively.

5.4.2 Temporal Graph Construction

To download the brain region atlases and to compute the mean brain activity of the brain regions over time the functions implemented in `nilearn` were used.

[1] `http://preprocessed-connectomes-project.org`

[2] Code available at `https://nilearn.github.io`

A probabilistic brain atlas due to [Var+11] with 39 regions of interest was used. To estimate the functional connectivity we follow [KYK21] and compute the brain regions' correlation within a time window of 36 seconds every 2.16 seconds. For fMRI data with 1.38Hz temporal resolution this results in an absolute window length $w = 50$ and offset $\Delta t = 3$. As a threshold to insert temporal edges the 30-th percentile of the brain regions functional connectivity $FC(t)$ was chosen for each time step t.

We publish the code to build the temporal graphs and the temporal graph data sets[3]. The data sets' statistics are listed in Table 5.1.

5.4.3 Experimental Protocol

Due to the size of fMRI data 100 subjects were randomly selected per data set. For classification we follow the methods of Section 4.2.3. Solely the hyperparameters with respect to the proximity methods differ.

The vertex features for temporal graph warping were computed for neighborhood distance $k \in [4]$ with weights $\alpha_k = 0.5^k$ and $\alpha_k = k^{-1}$. Their distances were measured by the L_1 and L_2 norm. The temporal random walk approximation kernel was computed for $S \in \{10^2, 10^3, 10^4\}$ length-k walks with $k \in [10]$. The Random Walk kernel on the static graph expansion could be computed for walk length $k \in [2]$ within a time limit of one hour per kernel computation (recall that the the temporal graphs' feature map have size $\mathcal{O}(|V|^k)$ for the number of vertices $|V|$).

The Directed Line Graph Expansion could not be computed on the brain data sets due to the memory consumption that is quadratic in the number of temporal edges. The computation of Weisfeiler-Lehmann subtree kernel and the temporal graphlet kernel failed on our data sets. It was not enough time in this work to adapt the code.

5.4.4 Results

The mean classification accuracy and standard deviation for our classification tasks depicted in Section 5.4.1 for each method with the best choice of hyperparameters is reported in Table 5.2. Random chance is 50% for the Autism and ADHD data set and 79% for the brain developement classification task.

Comparing the brain regions' activity signals using dynamic time warping does not help to classify patients with Autism or ADHD but children and adults. While brain regions' connectivity over the whole signal's period of time classifies adults and children best, it can not classify Autism or ADHD patients better than chance. Comparing dynamic functional brain regions' connectivity using temporal brain networks performs as well as comparing the brain regions' activity signals on the brain development classification task. It slightly improves the classification accuracy for the Autism and ADHD discrimination task. The classification results for all parameter choices appended in Chapter 7 show that the results vary significantly with respect to the parameters. Considering the high variance our approach does not seem to be a robust model to detect Autism or ADHD.

[3]Available at `https://github.com/jonasschultemattler/cmptgraphs`

Table 5.2: Classification accuracy in % for the tasks depicted in Section 5.4.1. We denote the brain regions' activity signals comparison using dynamic time warping by dtw, and using the regions' activity correlation by corr. Temporal graph warping is abbreviated with tgw and temporal graph kernel with tkernel. Thereby, the vertex signatures are reported. Random Walk kernel on temporal graph static expansion is called SE-RW and the Random Walk Approximation kernel Approx.

Algorithm		**Data set**		
		AUTISM	DEVELOPMENT	ADHD
dtw		52.0±$_{7.5}$	92.0±$_{9.3}$	52.5±$_{5.0}$
corr		56.4±$_{11.6}$	**93.8**±$_{5.1}$	42.5±$_{10.0}$
tgw	interaction	57.0±$_{9.3}$	89.0±$_{8.6}$	45.0±$_{6.1}$
tgw	degree	52.0±$_{10.8}$	88.0±$_{8.1}$	47.5±$_{5.0}$
tkernel	SE-RW	56.4±$_{10.9}$	92.0±$_{4.0}$	40.0±$_{14.6}$
tkernel	Approx.	**71.0**±$_{10.2}$	85.0±$_{3.2}$	**62.5**±$_{7.9}$

Table 5.3: Correctly matched brain regions using dtgw in %.

Signature		**Data set**		
		AUTISM	DEVELOPMENT	ADHD
interaction	∅	49.6	49.1	44.4
interaction	max	94.9	89.7	89.7
degree	∅	48.7	48.8	44.5
degree	max	87.2	89.7	89.7

We try to investigate our approach by comparing brain regions dynamic functional connectivity in the following chapter.

5.5 Functional Decoding using DTGW

Knowing a matching between the vertices, i.e., brain regions, we can try to detect brain regions by their functional connectivity using dynamic temporal graph warping, see Section 3.2.1, to match the most similar brain regions over time

If they are matched correctly, we can identify brain regions by their dynamic functional connectivity. This would also proof our concept to discriminate patients by their brain regions' connectivity using temporal graph warping. Otherwise we can conclude that either the extracted functional connectivity or the vertex signatures are not expressive for the brain regions.

Table 5.3 reports the proportion of correctly matched brain regions per vertex signature and data set of the control group (in the case of brain development we considered the adults). Therein, the highest average proportion of correctly matched regions per signature (for any depth or distance norm) is reported. Not more than half of the brain regions were matched correctly – using the interaction vertex feature or the degree feature. This indicates that either the brain regions' features or the extracted functional connectivity do not contain enough information to detect the brain regions.

With that method at hand brain regions with most similar functional connectivity over time can be identified. One application could be to compare brain atlases or the role of brain regions between different patients or under certain tasks.

5.6 Discussion

We found that proximity measures of temporal graphs can be applied to fMRI data in order to classify brain connectomes by their dynamic functional connectivity. Despite temporal graphs are an appropriate model, we did not find a robust method to identify autism, ADHD, or children better than static connectivity comparison techniques.

This can be due to various reasons: We do not know how well the built temporal graphs capture the brain regions' dynamic functional connectivity. Is a binary connection of brain regions too simple? Do the graph comparison methods account for the differences in dynamic connections that are indicative for distinct brain connectivity?

This work presents a first simple approach to classify brains by modeling them as temporal graphs and provides a framework for future research. Comparing brains modeled as temporal graphs might be a viable way as they do not make simplifications as static graph analysis. Maybe the dynamics in brain connectivity are indicative for brain function and diseases. With expertise in neuroscience future hypotheses-driven work might yield a way to discriminate patients with brain diseases or reveal differences in the brain regions' functions by comparing temporal graphs that model their dynamic connectivity.

6 Conclusion

We wish to conclude this work with a guide to select an appropriate method when faced with the task to compare temporal graphs and look into future work.

Guide. The evaluated concepts measure temporal graph proximity in three different ways. The temporal graph kernels measure similarity by the number of common temporal walks or common small subgraphs. Dynamic temporal graph warping measures temporal graphs' distance by the difference between most common vertices over time. The method to choose depends on the application and the temporal graphs' properties.

If a vertex matching is reasonable or even interesting, dynamic temporal graph warping is the method to choose. It allows to use application specific knowledge to design vertex features and offers an explanation for the proximity score. Yet, a pairwise comparison reaches its limits when the number of graphs to compare gets big. Also comparing temporal graphs with thousands of vertices might be too expensive.

Temporal graph kernels capture common features of the temporal graphs. As the running time is dominated by computing the temporal graphs' features, it allows a quadratically faster comparison with respect to the number of temporal graphs. When using a temporal walk kernel note that the number of distinct colored temporal walks grows exponentially in the walk length. To compare vertex labeled temporal graphs with a big vertex label alphabet or unlabeled temporal graphs the temporal walk kernel approximation variant that samples a fixed number of random temporal walks offers a desirable trade off. The temporal graphlet kernel is efficiently computable for temporal graphs with thousands of vertices and temporal edges and performed best on classifying dissemination processes. Yet, it remains to be adapted to non binary vertex labeled temporal graphs and evaluated on other data sets.

Future work. It will be exciting to compare temporal graphs in other applications and further explore the comparison methods' strengths and weaknesses. These could e.g. be dynamic biomolecular interactions networks such as dynamic genes, protein, or molecule networks. In neuroscience the comparison of temporal graphs that model dynamic functional brain connectivity has critical applications. It seems rewarding to follow research in that direction.

Therefore, it is compelling to develop further comparison algorithms for temporal graphs. Two questions could be pursued further:

Can other established static graph kernels be lifted to the temporal case? Can dynamic temporal graph warping be modified in a fruitful way? The restriction of a fixed vertex matching might not be suited for certain applications in which the role of vertices change over time. Instead a warping path can be computed based on the temporal layers' similarity e.g. using static graph kernels. Besides, Dynamic Time Warping could be substituted with other time series comparison methods e.g. Move-Split-Merge [SAD13]. Besides, time series comparison techniques can be discarded and just a vertex assignment be computed based on vertex features that incorporate temporal information. Vertex embedding techniques for temporal graph learning could be used.

7 Appendix

Experimental Results for Classifying Dissemination Processes

Identification Task

Table 7.1: Classification accuracy in % for the identification task using temporal graph kernels for all parameter choices. We report the accuracy of the best classifier. The temporal graph static expansion is abbreviated with SE, directed line graph expansion with LG, random walk kernel with RW, Weisfeiler-Lehmann kerrnel with WL, and temporal graphlet kernel with TGK. The $\star$ symbolizes star-graphlets and $\wedge$ triangle-graphlets. The parameter k is the walk length, subtree height, or the graphlets' number of vertices respectively.

Kernel		**Data set**					
	k	MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
SE-RW	1	**92.8**±$_{4.5}$	95.2±$_{5.1}$	94.2±$_{4.8}$	89.6±$_{6.3}$	98.0±$_{2.6}$	93.4±$_{5.0}$
SE-RW	2	89.5±$_{6.5}$	94.0±$_{4.7}$	92.6±$_{6.5}$	88.8±$_{6.3}$	95.8±$_{4.8}$	90.8±$_{5.9}$
SE-RW	3	87.6±$_{6.4}$	**95.8**±$_{4.2}$	93.0±$_{5.1}$	86.6±$_{6.4}$	95.6±$_{4.4}$	90.4±$_{6.7}$
SE-WL	1	91.1±$_{5.6}$	94.2±$_{4.8}$	91.4±$_{6.6}$	88.4±$_{8.9}$	96.6±$_{3.2}$	92.6±$_{4.4}$
SE-WL	2	90.9±$_{6.1}$	94.0±$_{3.7}$	94.0±$_{6.1}$	87.0±$_{8.9}$	92.4±$_{4.2}$	94.0±$_{6.5}$
SE-WL	3	88.6±$_{5.6}$	93.4±$_{6.4}$	92.6±$_{5.8}$	87.2±$_{7.5}$	92.2±$_{4.6}$	89.8±$_{7.1}$
LG-RW	1	91.5±$_{4.6}$	87.0±$_{11.4}$	89.0±$_{8.7}$	87.8±$_{7.3}$	94.6±$_{5.0}$	91.8±$_{6.2}$
LG-RW	2	89.4±$_{5.4}$	90.4±$_{8.3}$	91.4±$_{7.3}$	82.4±$_{7.1}$	93.8±$_{5.5}$	86.4±$_{7.9}$
LG-WL	1	91.5±$_{5.6}$	91.6±$_{7.2}$	94.8±$_{6.3}$	**92.0**±$_{5.1}$	**99.0**±$_{2.0}$	93.4±$_{5.9}$
LG-WL	2	86.1±$_{5.5}$	92.0±$_{7.0}$	94.4±$_{5.2}$	90.0±$_{5.5}$	97.2±$_{2.8}$	91.0±$_{6.6}$
LG-WL	3	83.2±$_{6.3}$	93.2±$_{6.4}$	94.8±$_{5.5}$	89.0±$_{7.3}$	97.2±$_{3.2}$	**95.0**±$_{5.5}$
TGK-$\star$	1	87.0±$_{6.9}$	89.2±$_{7.8}$	94.0±$_{5.9}$	78.4±$_{8.1}$	98.0±$_{2.4}$	87.8±$_{7.9}$
TGK-$\star$	2	88.6±$_{7.2}$	91.6±$_{7.9}$	94.2±$_{5.9}$	79.6±$_{8.4}$	97.0±$_{2.4}$	87.4±$_{7.7}$
TGK-$\star$	3	87.6±$_{6.3}$	89.8±$_{8.8}$	97.0±$_{2.4}$	78.6±$_{9.1}$	98.0±$_{2.4}$	87.2±$_{7.7}$
TGK-all	1	86.6±$_{7.5}$	88.8±$_{8.2}$	95.0±$_{4.5}$	76.6±$_{8.4}$	98.0±$_{2.4}$	88.0±$_{8.1}$
TGK-all	2	88.8±$_{7.3}$	91.4±$_{7.9}$	**97.0**±$_{2.4}$	79.8±$_{9.1}$	97.0±$_{4.0}$	87.6±$_{8.8}$
TGK-all	3	87.6±$_{6.3}$	89.6±$_{9.2}$	97.0±$_{2.4}$	77.6±$_{10.3}$	96.0±$_{3.7}$	88.2±$_{8.5}$
TGK-$\wedge$	1	90.7±$_{6.2}$	94.0±$_{2.0}$	92.4±$_{6.2}$	89.0±$_{9.2}$	97.0±$_{4.0}$	93.0±$_{4.0}$
TGK-$\wedge$	2	91.3±$_{5.8}$	92.2±$_{7.1}$	95.0±$_{6.3}$	88.6±$_{9.7}$	97.0±$_{2.4}$	91.0±$_{5.8}$
TGK-$\wedge$	3	91.3±$_{7.2}$	92.8±$_{6.4}$	92.0±$_{7.1}$	90.0±$_{7.1}$	96.0±$_{4.1}$	90.6±$_{5.6}$

Table 7.2: Classification accuracy in % for the identification task using dtgw for all parameter choices. We report the accuracy of the best classifier for subtree and walk vertex features. The walk length or subtree height is k, the distance function is d, and the vertex "deletion" cost C_D for a vertex is either zero, the norm of feature f, or the feature norm divided by the number of unmatched vertices f_N.

	Variant			**Data set**					
	k	d	C_D	MIT	HIGHSCHOOL	INFECTIOUS	TUMBLR	DBPL	FACEBOOK
subtree	0	L_1	0	93.7±6.1	87.0±10.3	92.0±4.0	55.0±6.3	97.0±4.0	74.0±10.2
	0	L_2	0	92.6±9.2	89.0±4.9	90.0±5.5	52.0±7.5	**99.0**±2.0	66.0±9.7
	0	L_1	f	93.7±6.1	85.0±4.5	92.0±4.0	61.0±8.6	97.0±2.4	67.0±8.7
	0	L_2	f	92.6±9.2	86.0±4.9	90.0±5.5	49.0±15.3	98.0±2.4	67.0±12.1
	0	L_1	f_N	93.7±6.1	86.0±4.9	92.0±4.0	55.0±13.0	97.0±2.4	76.0±10.2
	0	L_2	f_N	92.6±9.2	85.0±3.2	90.0±5.5	**63.0**±12.9	98.0±2.4	70.0±4.5
	1	L_1	0	92.8±2.3	87.0±6.8	90.0±4.5	60.0±8.4	**99.0**±2.0	68.0±6.0
	1	L_2	0	93.7±5.2	**92.0**±2.4	88.0±5.1	56.0±5.8	98.0±4.0	72.0±8.7
	1	L_1	f	92.8±2.3	87.0±4.0	90.0±4.5	56.0±3.7	**99.0**±2.0	77.0±5.1
	1	L_2	f	93.7±5.2	87.0±6.0	88.0±5.1	58.0±13.6	98.0±2.4	76.0±6.6
	1	L_1	f_N	92.8±2.3	88.0±4.0	90.0±4.5	57.0±9.8	96.0±2.0	74.0±6.6
	1	L_2	f_N	93.7±5.2	81.0±13.2	88.0±5.1	48.0±9.3	97.0±4.0	69.0±9.7
	2	L_1	0	**95.9**±5.0	91.0±8.6	**96.0**±3.7	59.0±3.7	98.0±2.4	74.0±8.0
	2	L_2	0	94.9±5.5	87.0±8.1	92.0±5.1	51.0±3.7	98.0±2.4	74.0±7.3
	2	L_1	f	**95.9**±5.0	87.0±4.0	**96.0**±3.7	49.0±3.7	**99.0**±2.0	78.0±7.5
	2	L_2	f	94.9±5.5	87.0±6.0	92.0±5.1	59.0±10.7	98.0±2.4	72.0±11.7
	2	L_1	f_N	**95.9**±5.0	83.0±4.0	**96.0**±3.7	55.0±6.3	96.0±3.7	**80.0**±8.4
	2	L_2	f_N	94.9±5.5	84.0±5.8	92.0±5.1	55.0±5.5	97.0±4.0	77.0±7.5
walk	1	L_1	0	68.1±6.9	56.0±8.6	85.0±5.5	53.0±9.3	84.0±8.0	69.0±5.8
	1	L_2	0	64.8±12.8	60.0±7.1	81.0±5.8	49.0±10.2	81.0±5.8	62.0±13.6
	1	L_1	f	68.1±6.9	76.0±3.7	85.0±5.5	59.0±8.0	76.0±13.2	68.0±9.8
	1	L_2	f	64.8±12.8	83.0±8.7	81.0±5.8	45.0±6.3	78.0±10.3	73.0±4.0
	1	L_1	f_N	68.1±6.9	60.0±10.0	85.0±5.5	55.0±8.4	83.0±2.4	73.0±15.0
	1	L_2	f_N	64.8±12.8	74.0±3.7	81.0±5.8	54.0±7.3	77.0±5.1	67.0±8.1
	2	L_1	0	67.8±15.6	61.0±5.8	83.0±8.7	55.0±4.5	79.0±10.2	77.0±5.1
	2	L_2	0	73.2±12.5	66.0±5.8	84.0±9.7	49.0±9.7	73.0±12.1	69.0±6.6
	2	L_1	f	67.8±15.6	73.0±10.3	83.0±8.7	63.0±12.9	79.0±8.0	73.0±9.8
	2	L_2	f	73.2±12.5	70.0±8.4	84.0±9.7	51.0±7.3	73.0±11.2	66.0±3.7
	2	L_1	f_N	67.8±15.6	64.0±19.3	83.0±8.7	45.0±11.4	80.0±9.5	73.0±6.0
	2	L_2	f_N	73.2±12.5	68.0±8.1	84.0±9.7	63.0±8.1	77.0±9.3	67.0±11.2

Table 7.3: Running times for the identification task in seconds.

Algorithm		Data set					
		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	$1.34 \cdot 10^{3}$	$2.51 \cdot 10^{1}$	$5.74 \cdot 10^{0}$	$1.38 \cdot 10^{1}$	$3.88 \cdot 10^{1}$	$3.65 \cdot 10^{1}$
	walks	$2.74 \cdot 10^{3}$	$4.35 \cdot 10^{1}$	$8.51 \cdot 10^{0}$	$1.94 \cdot 10^{1}$	$7.53 \cdot 10^{0}$	$1.29 \cdot 10^{2}$
tkernel	SE	$7.59 \cdot 10^{-1}$	$2.04 \cdot 10^{-1}$	$1.43 \cdot 10^{-1}$	$4.1 \cdot 10^{-2}$	$5.1 \cdot 10^{-2}$	$2.28 \cdot 10^{-1}$
	LG	$2.47 \cdot 10^{2}$	$1.22 \cdot 10^{0}$	$4.13 \cdot 10^{-1}$	$8.1 \cdot 10^{-2}$	$2.15 \cdot 10^{-1}$	$6.9 \cdot 10^{-2}$
	TGK	$1.01 \cdot 10^{0}$	$1.03 \cdot 10^{-1}$	$8.7 \cdot 10^{-2}$	$9.3 \cdot 10^{-2}$	$1.55 \cdot 10^{-1}$	$8.8 \cdot 10^{-2}$

Table 7.4: Classification accuracy in % for the identification task.

Algorithm		Data set					
		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	**95.9**$\pm_{5.0}$	92.0$\pm_{2.4}$	96.0$\pm_{3.7}$	63.0$\pm_{12.9}$	**99.0**$\pm_{2.0}$	80.0$\pm_{8.4}$
	walks	73.2$\pm_{12.5}$	83.0$\pm_{8.7}$	87.0$\pm_{6.8}$	63.0$\pm_{8.1}$	84.0$\pm_{8.0}$	77.0$\pm_{5.1}$
tkernel	SE	92.8$\pm_{4.5}$	**95.8**$\pm_{4.2}$	94.2$\pm_{4.8}$	89.6$\pm_{6.3}$	98.0$\pm_{2.6}$	94.0$\pm_{6.5}$
	LG	91.5$\pm_{5.6}$	93.2$\pm_{6.4}$	94.8$\pm_{6.3}$	**92.0**$\pm_{5.1}$	**99.0**$\pm_{2.0}$	**95.0**$\pm_{5.5}$
	TGK	91.3$\pm_{7.2}$	94.0$\pm_{2.0}$	**97.0**$\pm_{2.4}$	90.0$\pm_{7.1}$	98.0$\pm_{2.4}$	93.0$\pm_{4.0}$

Table 7.5: Classification accuracy in % for the identification task on data sets that contain temporal graphs whose proportion of infected vertices is 50%.

Algorithm		Data set					
		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	**95.9**$\pm_{5.0}$	90.0$\pm_{3.2}$	97.0$\pm_{2.4}$	71.0$\pm_{6.6}$	**100.0**$\pm_{0.0}$	84.0$\pm_{3.7}$
	walks	73.2$\pm_{12.5}$	80.0$\pm_{4.5}$	90.0$\pm_{6.3}$	67.0$\pm_{11.7}$	90.0$\pm_{6.3}$	89.0$\pm_{5.8}$
tkernel	SE	92.8$\pm_{4.5}$	**98.6**$\pm_{2.2}$	97.0$\pm_{4.3}$	**100.0**$\pm_{0.0}$	98.2$\pm_{2.7}$	**100.0**$\pm_{0.0}$
	LG	91.5$\pm_{5.6}$	97.8$\pm_{2.7}$	**97.6**$\pm_{2.5}$	100.0$\pm_{0.0}$	100.0$\pm_{0.0}$	100.0$\pm_{0.0}$
	TGK	91.3$\pm_{7.2}$	97.0$\pm_{2.8}$	97.4$\pm_{3.8}$	100.0$\pm_{0.0}$	99.0$\pm_{2.0}$	100.0$\pm_{0.0}$

Differentiation Task

Table 7.6: Classification accuracy in % for the differentiation task using temporal graph kernels for all parameter choices. We report the accuracy of the best classifier. The temporal graph static expansion is abbreviated with SE, directed line graph expansion with LG, random walk kernel with RW, Weisfeiler-Lehmann kerrnel with WL, and temporal graphlet kernel with TGK. The $\star$ symbolizes star-graphlets and $\wedge$ triangle-graphlets. The parameter k is the walk length, subtree height, or the graphlets' number of vertices respectively.

Kernel		**Data set**					
	k	MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
SE-RW	1	55.9±8.3	88.8±5.9	71.0±15.2	74.6±8.4	81.4±8.5	81.8±7.0
SE-RW	2	57.7±8.5	89.4±7.5	86.4±10.8	74.0±11.9	78.6±8.3	79.0±8.7
SE-RW	3	67.3±9.1	87.2±8.0	75.0±10.5	75.0±12.6	79.6±6.3	80.8±6.9
SE-WL	1	55.6±8.7	**91.6**±5.4	75.0±10.5	71.4±10.9	83.4±7.8	80.4±7.9
SE-WL	2	57.4±8.9	89.8±6.6	66.0±12.4	74.0±11.8	76.4±8.3	74.8±8.6
SE-WL	3	58.3±8.3	89.2±6.5	72.0±14.4	73.0±10.1	75.6±8.9	75.0±7.1
LG-RW	1	**84.2**±7.5	89.6±6.7	**87.4**±9.1	76.0±11.4	81.8±9.4	80.0±11.6
LG-RW	2	79.4±11.4	90.6±7.1	83.6±8.6	73.0±10.4	78.0±5.1	76.0±10.2
LG-WL	1	58.0±9.8	88.8±7.0	65.0±7.1	75.6±12.9	80.0±8.4	81.2±8.0
LG-WL	2	58.0±10.4	85.6±8.5	71.6±12.6	74.0±14.1	81.0±10.7	75.2±8.7
LG-WL	3	63.0±8.4	83.0±8.4	71.0±8.6	73.0±13.8	77.0±7.5	73.8±8.4
TGK-$\star$	1	77.1±8.2	90.0±5.1	86.8±10.2	72.0±12.1	84.0±8.7	77.0±7.5
TGK-$\star$	2	80.7±9.7	90.6±5.6	86.4±10.2	72.0±7.5	83.6±8.2	79.4±10.0
TGK-$\star$	3	81.7±9.2	90.4±5.6	87.0±10.0	70.8±11.6	84.4±8.9	80.2±8.6
TGK-all	1	79.7±7.7	89.6±4.7	84.4±10.9	75.0±9.4	84.6±8.6	**82.2**±8.7
TGK-all	2	79.0±10.2	90.6±5.6	85.6±10.3	73.2±9.5	87.2±7.2	79.2±9.1
TGK-all	3	81.9±8.8	90.8±5.2	85.4±10.8	72.0±8.1	87.2±5.9	80.8±6.9
TGK-$\wedge$	1	72.3±10.4	87.0±8.3	78.8±13.3	78.6±9.7	89.4±6.7	76.2±9.4
TGK-$\wedge$	2	74.2±7.3	89.6±6.5	78.8±14.0	80.6±8.2	89.8±7.2	78.0±5.1
TGK-$\wedge$	3	72.5±8.3	90.4±6.1	79.0±5.8	**81.0**±8.6	**90.2**±7.8	75.4±8.9

Table 7.7: Classification accuracy in % for the differentiation task using dtgw for all parameter choices. We report the accuracy of the best classifier for subtree and walk vertex features. The walk length or subtree height is k, the distance function is d, and the vertex "deletion" cost C_D for a vertex is either zero, the norm of feature f, or the feature norm divided by the number of unmatched vertices f_N.

	Variant			**Data set**					
	k	d	C_D	MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
subtree	0	L_1	0	64.8$\pm_{12.5}$	84.0$\pm_{7.3}$	72.0$\pm_{4.0}$	54.0$\pm_{8.6}$	64.0$\pm_{4.9}$	71.0$\pm_{9.7}$
	0	L_2	0	59.0$\pm_{11.7}$	80.0$\pm_{6.3}$	74.0$\pm_{5.8}$	51.0$\pm_{8.6}$	64.0$\pm_{12.4}$	71.0$\pm_{7.3}$
	0	L_1	f	64.8$\pm_{12.5}$	79.0$\pm_{8.0}$	72.0$\pm_{4.0}$	67.0$\pm_{6.0}$	65.0$\pm_{7.7}$	74.0$\pm_{9.2}$
	0	L_2	f	59.0$\pm_{11.7}$	77.0$\pm_{6.8}$	74.0$\pm_{5.8}$	60.0$\pm_{10.5}$	66.0$\pm_{9.2}$	75.0$\pm_{3.2}$
	0	L_1	f_N	64.8$\pm_{12.5}$	76.0$\pm_{7.3}$	72.0$\pm_{4.0}$	59.0$\pm_{4.9}$	63.0$\pm_{9.3}$	71.0$\pm_{6.6}$
	0	L_2	f_N	59.0$\pm_{11.7}$	75.0$\pm_{6.3}$	74.0$\pm_{5.8}$	53.0$\pm_{8.1}$	59.0$\pm_{13.2}$	75.0$\pm_{8.4}$
	1	L_1	0	59.8$\pm_{12.1}$	83.0$\pm_{6.8}$	78.0$\pm_{7.5}$	45.0$\pm_{12.6}$	60.0$\pm_{4.5}$	71.0$\pm_{5.8}$
	1	L_2	0	**66.1**$\pm_{4.7}$	**87.0**$\pm_{4.0}$	**78.0**$\pm_{6.8}$	48.0$\pm_{9.3}$	63.0$\pm_{5.1}$	74.0$\pm_{5.8}$
	1	L_1	f	59.8$\pm_{12.1}$	77.0$\pm_{13.3}$	78.0$\pm_{7.5}$	62.0$\pm_{12.9}$	63.0$\pm_{5.1}$	75.0$\pm_{8.9}$
	1	L_2	f	**66.1**$\pm_{4.7}$	73.0$\pm_{8.1}$	**78.0**$\pm_{6.8}$	66.0$\pm_{10.7}$	71.0$\pm_{5.8}$	76.0$\pm_{8.6}$
	1	L_1	f_N	59.8$\pm_{12.1}$	70.0$\pm_{3.2}$	78.0$\pm_{7.5}$	59.0$\pm_{9.7}$	65.0$\pm_{4.5}$	66.0$\pm_{5.8}$
	1	L_2	f_N	**66.1**$\pm_{4.7}$	69.0$\pm_{8.0}$	**78.0**$\pm_{6.8}$	62.0$\pm_{13.3}$	65.0$\pm_{8.4}$	71.0$\pm_{8.6}$
	2	L_1	0	65.1$\pm_{9.4}$	84.0$\pm_{8.0}$	77.0$\pm_{9.3}$	48.0$\pm_{16.0}$	69.0$\pm_{10.2}$	72.0$\pm_{7.5}$
	2	L_2	0	63.9$\pm_{12.0}$	87.0$\pm_{6.8}$	72.0$\pm_{9.3}$	51.0$\pm_{9.7}$	69.0$\pm_{8.0}$	73.0$\pm_{9.3}$
	2	L_1	f	65.1$\pm_{9.4}$	75.0$\pm_{8.9}$	77.0$\pm_{9.3}$	**68.0**$\pm_{2.4}$	65.0$\pm_{6.3}$	73.0$\pm_{9.3}$
	2	L_2	f	63.9$\pm_{12.0}$	78.0$\pm_{9.3}$	72.0$\pm_{9.3}$	62.0$\pm_{12.1}$	**72.0**$\pm_{4.0}$	72.0$\pm_{2.4}$
	2	L_1	f_N	65.1$\pm_{9.4}$	73.0$\pm_{8.7}$	77.0$\pm_{9.3}$	57.0$\pm_{6.8}$	63.0$\pm_{9.3}$	**79.0**$\pm_{2.0}$
	2	L_2	f_N	63.9$\pm_{12.0}$	73.0$\pm_{9.3}$	72.0$\pm_{9.3}$	64.0$\pm_{7.3}$	66.0$\pm_{7.3}$	75.0$\pm_{5.5}$
walk	1	L_1	0	54.7$\pm_{11.4}$	59.0$\pm_{3.7}$	59.0$\pm_{10.2}$	60.0$\pm_{13.8}$	64.0$\pm_{10.7}$	56.0$\pm_{7.3}$
	1	L_2	0	55.7$\pm_{7.6}$	48.0$\pm_{16.0}$	62.0$\pm_{5.1}$	52.0$\pm_{4.0}$	59.0$\pm_{5.8}$	53.0$\pm_{7.5}$
	1	L_1	f	54.7$\pm_{11.4}$	59.0$\pm_{7.3}$	59.0$\pm_{10.2}$	54.0$\pm_{7.3}$	60.0$\pm_{8.9}$	60.0$\pm_{6.3}$
	1	L_2	f	55.7$\pm_{7.6}$	59.0$\pm_{11.1}$	62.0$\pm_{5.1}$	48.0$\pm_{7.5}$	64.0$\pm_{10.2}$	68.0$\pm_{7.5}$
	1	L_1	f_N	54.7$\pm_{11.4}$	54.0$\pm_{9.7}$	59.0$\pm_{10.2}$	61.0$\pm_{4.9}$	62.0$\pm_{9.3}$	60.0$\pm_{8.4}$
	1	L_2	f_N	55.7$\pm_{7.6}$	57.0$\pm_{6.8}$	62.0$\pm_{5.1}$	57.0$\pm_{6.8}$	62.0$\pm_{10.3}$	67.0$\pm_{4.0}$
	2	L_1	0	48.6$\pm_{5.2}$	54.0$\pm_{13.2}$	57.0$\pm_{8.7}$	59.0$\pm_{6.6}$	58.0$\pm_{15.0}$	59.0$\pm_{14.6}$
	2	L_2	0	52.5$\pm_{5.5}$	64.0$\pm_{10.2}$	53.0$\pm_{7.5}$	57.0$\pm_{10.3}$	54.0$\pm_{8.0}$	62.0$\pm_{14.4}$
	2	L_1	f	48.6$\pm_{5.2}$	68.0$\pm_{10.3}$	57.0$\pm_{8.7}$	50.0$\pm_{12.6}$	64.0$\pm_{12.4}$	69.0$\pm_{11.1}$
	2	L_2	f	52.5$\pm_{5.5}$	63.0$\pm_{8.7}$	53.0$\pm_{7.5}$	54.0$\pm_{6.6}$	60.0$\pm_{7.1}$	53.0$\pm_{9.3}$
	2	L_1	f_N	48.6$\pm_{5.2}$	56.0$\pm_{9.7}$	57.0$\pm_{8.7}$	52.0$\pm_{12.1}$	55.0$\pm_{14.1}$	60.0$\pm_{5.5}$
	2	L_2	f_N	52.5$\pm_{5.5}$	69.0$\pm_{13.6}$	53.0$\pm_{7.5}$	59.0$\pm_{3.7}$	52.0$\pm_{13.6}$	67.0$\pm_{10.8}$

Table 7.8: Running times for the differentiation task in seconds.

Algorithm		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	$1.13 \cdot 10^3$	$6.64 \cdot 10^1$	$4.8 \cdot 10^0$	$8.04 \cdot 10^1$	$8.35 \cdot 10^1$	$9.95 \cdot 10^1$
	walks	$1.3 \cdot 10^3$	$8.88 \cdot 10^1$	$8.12 \cdot 10^0$	$3.83 \cdot 10^1$	$1.91 \cdot 10^1$	$6.12 \cdot 10^1$
tkernel	SE	$6.46 \cdot 10^0$	$2.01 \cdot 10^{-1}$	$2.53 \cdot 10^0$	$4.1 \cdot 10^{-2}$	$1.01 \cdot 10^{-1}$	$1.12 \cdot 10^{-1}$
	LG	$2.88 \cdot 10^2$	$2.91 \cdot 10^1$	$9.09 \cdot 10^0$	$1.46 \cdot 10^{-1}$	$1.99 \cdot 10^{-1}$	$8.2 \cdot 10^{-2}$
	TGK	$4.72 \cdot 10^{-1}$	$1 \cdot 10^{-1}$	$9.5 \cdot 10^{-2}$	$7.6 \cdot 10^{-2}$	$6.69 \cdot 10^{-1}$	$1.75 \cdot 10^{-1}$

Table 7.9: Classification accuracy in % for the differentiation task.

Algorithm		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	66.1$\pm$4.7	87.0$\pm$4.0	80.0$\pm$5.5	68.0$\pm$2.4	72.0$\pm$4.0	79.0$\pm$2.0
	walks	55.7$\pm$7.6	69.0$\pm$13.6	64.0$\pm$8.0	61.0$\pm$4.9	64.0$\pm$10.2	69.0$\pm$11.1
tkernel	SE	67.3$\pm$9.1	**91.6**$\pm$5.4	86.4$\pm$10.8	75.0$\pm$12.6	83.4$\pm$7.8	81.8$\pm$7.0
	LG	**84.2**$\pm$7.5	90.6$\pm$7.1	**87.4**$\pm$9.1	76.0$\pm$11.4	81.8$\pm$9.4	81.2$\pm$8.0
	TGK	81.9$\pm$8.8	91.2$\pm$7.7	87.0$\pm$10.0	**81.0**$\pm$8.6	**90.2**$\pm$7.8	**82.2**$\pm$8.7

Table 7.10: Classification accuracy in % for the differentiation task on data sets that contain temporal graphs whose proportion of infected vertices is 50%.

Algorithm		MIT	Highschool	Infectious	Tumblr	DBPL	Facebook
dtgw	subtrees	66.1$\pm$4.7	87.0$\pm$10.3	80.0$\pm$7.1	72.0$\pm$9.3	62.0$\pm$7.5	70.0$\pm$8.9
	walks	55.7$\pm$7.6	67.0$\pm$8.7	65.0$\pm$8.9	70.0$\pm$7.7	66.0$\pm$13.2	72.0$\pm$6.8
tkernel	SE	67.3$\pm$9.1	89.2$\pm$5.9	83.2$\pm$8.5	82.2$\pm$8.1	85.0$\pm$8.2	**83.6**$\pm$7.6
	LG	63.0$\pm$8.4	90.6$\pm$7.1	**83.6**$\pm$8.6	77.2$\pm$9.4	85.8$\pm$6.7	79.0$\pm$9.7
	TGK	**81.9**$\pm$8.8	**91.4**$\pm$6.2	80.0$\pm$3.2	**82.6**$\pm$8.4	**92.6**$\pm$5.7	79.0$\pm$10.0

Experimental Results for Classifying Brain Connectomes

Table 7.11: Classification accuracy in % using temporal graph warping.

Variant			Data set		
	k	d	Autism	Development	ADHD
interaction	1	L_1	56.0±7.3	85.0±4.5	40.0±14.6
interaction	1	L_2	50.0±4.5	**89.0**±8.6	45.0±20.3
interaction	2	L_1	**57.0**±9.3	88.0±4.0	40.0±12.2
interaction	2	L_2	53.0±6.8	84.0±2.0	45.0±6.1
interaction	3	L_1	45.0±3.2	87.0±8.7	40.0±14.6
interaction	3	L_2	54.0±8.6	88.0±2.4	42.5±20.3
degree	1	L_1	42.0±10.8	79.0±10.2	37.5±15.8
degree	1	L_2	47.0±11.7	82.0±4.0	35.0±9.4
degree	2	L_1	52.0±10.8	87.0±8.1	**47.5**±16.6
degree	2	L_2	43.0±6.8	85.0±4.5	47.5±5.0
degree	3	L_1	49.0±8.6	82.0±6.0	47.5±12.2
degree	3	L_2	43.0±10.3	88.0±8.1	47.5±21.5

Table 7.12: Classification accuracy in % using temporal graph kernels.

Kernel			Data set		
Variant	k	S	Autism	Development	ADHD
SE-RW	1		56.0±7.3	**92.0**±4.0	40.0±14.6
SE-RW	2		56.4±10.9	91.0±6.2	38.5±17.4
Approx	2	$1 \cdot 10^2$	42.0±5.1	79.0±10.2	45.5±15.9
Approx	2	$1 \cdot 10^3$	55.2±8.8	79.0±10.2	50.0±23.7
Approx	2	$1 \cdot 10^4$	54.6±8.6	81.6±10.6	55.0±13.7
Approx	3	$1 \cdot 10^2$	**71.0**±10.2	79.0±10.2	34.5±11.1
Approx	3	$1 \cdot 10^3$	51.4±12.0	79.0±4.9	**62.5**±7.9
Approx	3	$1 \cdot 10^4$	58.0±8.7	80.0±10.9	52.5±12.2
Approx	5	$1 \cdot 10^2$	43.0±10.6	79.0±10.2	52.5±16.6
Approx	5	$1 \cdot 10^3$	52.0±10.3	80.0±4.5	41.5±15.5
Approx	5	$1 \cdot 10^4$	58.0±2.4	83.0±8.1	52.0±14.2
Approx	10	$1 \cdot 10^2$	55.0±5.5	79.0±10.2	50.0±7.9
Approx	10	$1 \cdot 10^3$	44.0±8.6	79.0±10.2	47.5±14.6
Approx	10	$1 \cdot 10^4$	48.0±8.1	80.0±10.5	40.0±9.4

Table 7.13: Running times in seconds.

Algorithm		**Data set**		
		Autism	development	adhd
tw		$4.45 \cdot 10^{0}$	$4.61 \cdot 10^{0}$	$8.5 \cdot 10^{-1}$
corr		$9.46 \cdot 10^{-2}$	$9.07 \cdot 10^{-2}$	$3.55 \cdot 10^{-2}$
tgw	interaction	$1.04 \cdot 10^{1}$	$4.2 \cdot 10^{0}$	$4.45 \cdot 10^{0}$
tgw	degree	$5.33 \cdot 10^{0}$	$9.87 \cdot 10^{0}$	$1.19 \cdot 10^{0}$
tkernel	SE	$2.18 \cdot 10^{2}$	$5.4 \cdot 10^{0}$	$2.27 \cdot 10^{0}$
tkernel	Approx.	$3.23 \cdot 10^{1}$	$3.01 \cdot 10^{1}$	$1.32 \cdot 10^{1}$

Literature

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993 (cit. on p. 9).

[ATK14] L. Akoglu, H. Tong, and D. Koutra. *Graph-based Anomaly Detection and Description: A Survey*. 2014. arXiv: 1404.4679 [cs.SI] (cit. on p. 5).

[Bel+17] P. Bellec, C. Chu, F. Chouinard-Decorte, Y. Benhajali, D. S. Margulies, and R. C. Craddock. "The Neuro Bureau ADHD-200 Preprocessed repository". In: *NeuroImage* 144 (2017). Data Sharing Part II, pp. 275–286. DOI: https://doi.org/10.1016/j.neuroimage.2016.06.034 (cit. on p. 36).

[Bel+20] M. Beladev, L. Rokach, G. Katz, I. Guy, and K. Radinsky. "TdGraphEmbed: Temporal Dynamic Graph-Level Embedding". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM '20. Virtual Event, Ireland: Association for Computing Machinery, 2020, 55–64. DOI: 10.1145/3340531.3411953 (cit. on p. 6).

[Cal+14] V. D. Calhoun, R. Miller, G. Pearlson, and T. Adalı. "The chronnectome: time-varying connectivity networks as the next frontier in fMRI data discovery". In: *Neuron* 84.2 (2014), pp. 262–274 (cit. on p. 31).

[Dak+19] N. Dakiche, F. Benbouzid-Si Tayeb, Y. Slimani, and K. Benatchba. "Tracking community evolution in social networks: A survey". In: *Information Processing & Management* 56.3 (2019), pp. 1084–1102. DOI: https://doi.org/10.1016/j.ipm.2018.03.005 (cit. on p. 5).

[DFC18] Y. Du, Z. Fu, and V. D. Calhoun. "Classification and Prediction of Brain Disorders Using Functional Connectivity: Promising but Challenging". In: *Frontiers in Neuroscience* 12 (2018). DOI: 10.3389/fnins.2018.00525 (cit. on pp. 31, 33).

[FKL19] F. V. Farahani, W. Karwowski, and N. R. Lighthall. "Application of Graph Theory for Identifying Connectivity Patterns in Human Brain Networks: A Systematic Review". In: *Frontiers in Neuroscience* 13 (2019). DOI: 10.3389/fnins.2019.00585 (cit. on p. 31).

[Fro+20] V. Froese, B. Jain, R. Niedermeier, and M. Renken. "Comparing temporal graphs using dynamic time warping". In: *Social Network Analysis and Mining* 10.1 (2020). DOI: 10.1007/s13278-020-00664-5 (cit. on pp. 6, 12, 13, 16, 20, 28, 29).

[HB04] B. Haasdonk and C. Bahlmann. "Learning with Distance Substitution Kernels". In: *Pattern Recognition*. Ed. by C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 220–227 (cit. on p. 24).

[Hos+22] M. M. Hosseinzadeh, M. Cannataro, P. H. Guzzi, and R. Dondi. "Temporal networks in biology and medicine: a survey on models, algorithms, and tools". In: *Network Modeling Analysis in Health Informatics and Bioinformatics* 12.1 (2022), p. 10. DOI: 10.1007/s13721-022-00406-x (cit. on p. 5).

[HS12] P. Holme and J. Saramäki. "Temporal networks". In: *Physics Reports* 519.3 (2012), pp. 97–125. DOI: 10.1016/j.physrep.2012.03.001 (cit. on p. 5).

[Hut+13] R. M. Hutchison, T. Womelsdorf, E. A. Allen, P. A. Bandettini, V. D. Calhoun, M. Corbetta, S. Della Penna, J. H. Duyn, G. H. Glover, J. Gonzalez-Castillo, D. A. Handwerker, S. Keilholz, V. Kiviniemi, D. A. Leopold, F. de Pasquale, O. Sporns, M. Walter, and C. Chang. "Dynamic functional connectivity: Promise, issues, and interpretations". In: *NeuroImage* 80 (2013). Mapping the Connectome, pp. 360–378. DOI: https://doi.org/10.1016/j.neuroimage.2013.05.079 (cit. on p. 31).

[JT09] S. Jouili and S. Tabbone. "Graph Matching Based on Node Signatures". In: *Graph-Based Representations in Pattern Recognition*. Ed. by A. Torsello, F. Escolano, and L. Brun. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 154–163 (cit. on p. 11).

[Kaz+20] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. *Representation Learning for Dynamic Graphs: A Survey*. 2020. arXiv: 1905.11485 [cs.LG] (cit. on p. 6).

[KGW16] N. M. Kriege, P.-L. Giscard, and R. Wilson. "On Valid Optimal Assignment Kernels and Applications to Graph Classification". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016 (cit. on p. 11).

[KJM20a] N. M. Kriege, F. D. Johansson, and C. Morris. "A survey on graph kernels". In: *Applied Network Science* 5.1 (2020). DOI: 10.1007/s41109-019-0195-3 (cit. on p. 5).

[KJM20b] N. M. Kriege, F. D. Johansson, and C. Morris. "A survey on graph kernels". In: *Applied Network Science* 5.1 (2020), p. 6. DOI: 10.1007/s41109-019-0195-3 (cit. on p. 31).

[KYK21] B.-H. Kim, J. C. Ye, and J.-J. Kim. *Learning Dynamic Graph Representation of Brain Connectome with Spatio-Temporal Attention*. 2021. arXiv: 2105.13495 [cs.CV] (cit. on pp. 31, 33, 37).

[Li+17] A. Li, S. P. Cornelius, Y.-Y. Liu, L. Wang, and A.-L. Barabási. "The fundamental advantages of temporal networks". In: *Science* 358.6366 (2017), pp. 1042–1046. DOI: 10.1126/science.aai7488. eprint: https://www.science.org/doi/pdf/10.1126/science.aai7488 (cit. on p. 5).

[Lur+20] D. J. Lurie, D. Kessler, D. S. Bassett, R. F. Betzel, M. Breakspear, S. Kheilholz, A. Kucyi, R. Liégeois, M. A. Lindquist, A. R. McIntosh, R. A. Poldrack, J. M. Shine, W. H. Thompson, N. Z. Bielczyk, L. Douw, D. Kraft, R. L. Miller, M. Muthuraman, L. Pasquini, A. Razi, D. Vidaurre, H. Xie, and V. D. Calhoun. "Questions and controversies in the study of time-varying functional connectivity in resting fMRI". In: *Network Neuroscience* 4.1 (Feb. 2020), pp. 30–69. DOI: `10.1162/netn_a_00116` (cit. on p. 31).

[Mic15] O. Michail. *An Introduction to Temporal Graphs: An Algorithmic Perspective.* 2015. arXiv: `1503.00278 [cs.DM]` (cit. on p. 14).

[Nie+13] J. Nielsen, B. Zielinski, P Fletcher, A. Alexander, N. Lange, E. Bigler, J. Lainhart, and J. Anderson. "Multisite functional connectivity MRI classification of autism: ABIDE results". In: *Frontiers in Human Neuroscience* 7 (2013). DOI: `10.3389/fnhum.2013.00599` (cit. on p. 36).

[Oet+a] L. Oettershagen, N. M. Kriege, C. Jordan, and P. Mutze. "A Temporal Graphlet Kernel For Classifying Dissemination in Evolving Networks". In: *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pp. 19–27. DOI: `10.1137/1.9781611977653.ch3`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611977653.ch3` (cit. on pp. 6, 12, 14, 15, 29).

[Oet+b] 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` (cit. on pp. 5, 6, 12, 14, 16, 19, 29, 35).

[OY+14] L. Ou-Yang, D.-Q. Dai, X.-L. Li, M. Wu, X.-F. Zhang, and P. Yang. "Detecting temporal protein complexes from dynamic protein-protein interaction networks". In: *BMC Bioinformatics* 15.1 (2014), p. 335. DOI: `10.1186/1471-2105-15-335` (cit. on p. 5).

[PBL17] A. Paranjape, A. R. Benson, and J. Leskovec. "Motifs in Temporal Networks". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining.* ACM, 2017. DOI: `10.1145/3018661.3018731` (cit. on pp. 5, 15).

[Ric+18] H. Richardson, G. Lisandrelli, A. Riobueno-Naylor, and R. Saxe. "Development of the social brain from age three to twelve years". In: *Nature Communications* 9.1 (2018), p. 1027. DOI: `10.1038/s41467-018-03399-2` (cit. on p. 36).

[SAD13] A. Stefan, V. Athitsos, and G. Das. "The Move-Split-Merge Metric for Time Series". In: *IEEE Transactions on Knowledge and Data Engineering* 25.6 (2013), pp. 1425–1438. DOI: `10.1109/TKDE.2012.88` (cit. on p. 41).

[San+22] C. P. Santana, E. A. de Carvalho, I. D. Rodrigues, G. S. Bastos, A. D. de Souza, and L. L. de Brito. "rs-fMRI and machine learning for ASD diagnosis: a systematic review and meta-analysis". In: *Scientific Reports* 12.1 (2022), p. 6030. DOI: 10.1038/s41598-022-09821-6 (cit. on p. 36).

[SC78] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. DOI: 10.1109/TASSP.1978.1163055 (cit. on p. 9).

[SGR19] U. Singer, I. Guy, and K. Radinsky. "Node Embedding over Temporal Graphs". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019. DOI: 10.24963/ijcai.2019/640 (cit. on p. 6).

[She+09] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. "Efficient graphlet kernels for large graph comparison". In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Ed. by D. van Dyk and M. Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 488–495 (cit. on p. 12).

[She+11] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. "Weisfeiler-Lehman Graph Kernels". In: *Journal of Machine Learning Research* 12.77 (2011), pp. 2539–2561 (cit. on p. 12).

[Tal+21] N. Talati, D. Jin, H. Ye, A. Brahmakshatriya, G. Dasika, S. Amarasinghe, T. Mudge, D. Koutra, and R. Dreslinski. "A Deep Dive Into Understanding The Random Walk-Based Temporal Graph Learning". In: *2021 IEEE International Symposium on Workload Characterization (IISWC)*. 2021, pp. 87–100. DOI: 10.1109/IISWC53511.2021.00019 (cit. on p. 6).

[TBF17] W. H. Thompson, P. Brantefors, and P. Fransson. "From static to temporal network theory: Applications to functional brain connectivity". In: *Network Neuroscience* 1.2 (June 2017), pp. 69–99. DOI: 10.1162/NETN_a_00011 (cit. on pp. 5, 6, 31).

[TF18] W. H. Thompson and P. Fransson. "A common framework for the problem of deriving estimates of dynamic functional brain connectivity". In: *NeuroImage* 172 (2018), pp. 896–902. DOI: https://doi.org/10.1016/j.neuroimage.2017.12.057 (cit. on p. 33).

[Var+11] G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, and B. Thirion. "Multi-subject dictionary learning to segment an atlas of brain spontaneous activity". In: *Lecture Notes in Computer Science*. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 562–573 (cit. on p. 37).

[Wei23] S. Wein. *Applications of Spatio-Temporal Graph Neural Network Models for Brain Connectivity Analysis*. 2023 (cit. on p. 31).