dg2pix: Pixel-Based Visual Analysis of Dynamic Graphs

Eren Cakmak[*] University of Konstanz
Dominik Jäckle[†] Independent Researcher
Tobias Schreck[‡] TU Graz
Daniel Keim[§] University of Konstanz

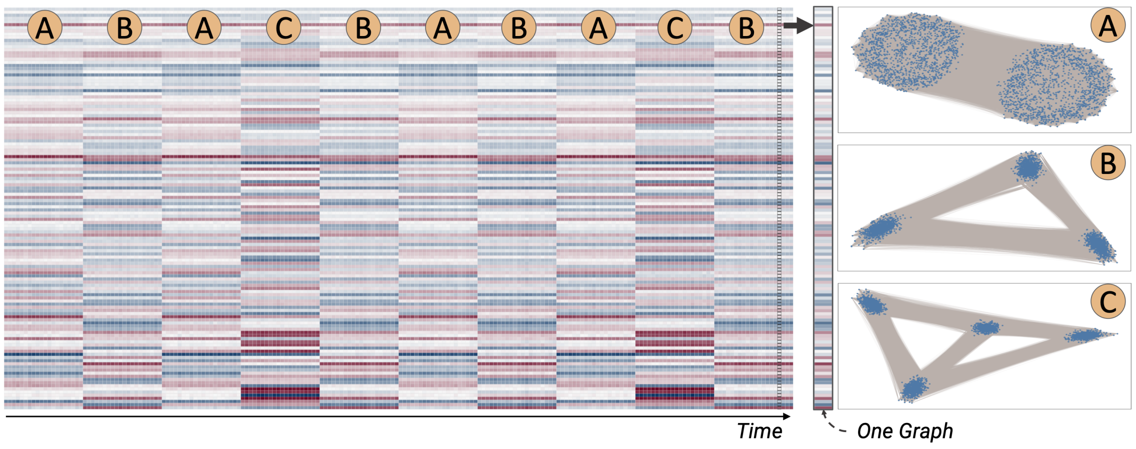


Figure 1: *dg2pix* provides an overview of temporal and structural changes in dynamic graphs. The example presents a synthetic dynamic graph (200-time steps) using graph2vec [35]. The x-axis presents the temporal dimension, and the y-axis displays for each time step a graph embedding as a pixel-bar. The reoccurring states (A-C) have for each time step the same amount of nodes (2500) and edges (350000) with a different number of clusters. Each state (20-time steps) was generated with SBM [25] with slight variations for the density of edges between clusters. The graphs (A-C) display a sample graph for each state. *dg2pix* enables us to explore and identify temporal changes, outlier graphs, and reoccurring graph structures at multiple temporal scales.

Abstract

Presenting long sequences of dynamic graphs remains challenging due to the underlying large-scale and high-dimensional data. We propose *dg2pix* , a novel pixel-based visualization technique, to visually explore temporal and structural properties in long sequences of large-scale graphs. The approach consists of three main steps: (1) the multiscale modeling of the temporal dimension; (2) unsupervised graph embeddings to learn low-dimensional representations of the dynamic graph data; and (3) an interactive pixel-based visualization to simultaneously explore the evolving data at different temporal aggregation scales. *dg2pix* provides a scalable overview of a dynamic graph, supports the exploration of long sequences of high-dimensional graph data, and enables the identification and comparison of similar temporal states. We show the applicability of the technique to synthetic and real-world datasets, demonstrating that temporal patterns in dynamic graphs can be identified and interpreted over time. *dg2pix* contributes a suitable intermediate representation between node-link diagrams at the high detail end and matrix representations on the low detail end.

Index Terms: Human-centered computing—Visualization—Visualization techniques; Machine learning—Learning paradigms—Unsupervised learning

[*]e-mail: eren.cakmak@uni-konstanz.de
[†]e-mail: dominikjaeckle@gmail.com
[‡]e-mail: tobias.schreck@cgv.tugraz.at
[§]e-mail: keim@uni-konstanz.de

1 Introduction

Dynamic graph visualizations are used in many real-world applications to present evolving relationships between entities, such as in social network analysis. A primary user task in such dynamic graph visualizations is to obtain an overview of the temporal dimension, for instance, to identify temporal states such as trends, outlier graphs, and similar graph structures over time [48]. However, visualizing large-scale dynamic graphs remains challenging as such visualization techniques have to present large amounts of evolving high-dimensional data in a readable and scalable manner [3].

Visualization techniques for dynamic graphs can be distinguished by the following primary categories: animation and timeline visualization [3]. However, both categories do not scale due to a large number of nodes, edges, and time steps [2]. Particularly, the evolving, potentially highly complex data may pose a significant challenge for the visual detection and traceability of changes in dynamic graph visualizations. Therefore, previous approaches for the visual analysis of dynamic graph data often rely on dimensionality reduction methods to provide an overview of higher-level structures over time [48]. Such dimensionality reduction methods reduce the complexity by embedding the evolving topological structures in low-dimensional space. While to date, some dynamic graph visualization techniques leverage dimensionality reduction methods (e.g., 2D embeddings [48]), they still fail to provide a scalable overview of the structural changes as the approaches depend on the temporal analysis scale and the designed feature vector (e.g., graph metrics).

We propose *dg2pix* (dynamic graph to pixel-based visualization), a novel visualization technique for large-scale dynamic graphs based on unsupervised graph learning methods (e.g., graph2vec [35] or FGSD [49]). The main goal is to provide a scalable overview of the temporal dimension and enable the initial exploration of the high-dimensional data to support the identification of temporal changes and similar temporal states. The visualization technique consists of three main steps: multiscale temporal modeling, graph embeddings, and a pixel-based visualization. The graph embedding reduces the dynamic graph to a low-dimensional representation (50-300 dimensions) and learns the similarity between graphs to capture the evolving topology of the high-dimensional data. The compact visualization technique allows users to interactively adapt the temporal analysis scale and compare high-level as well as fine-grained structural changes. We demonstrate the usefulness of our approach through two use cases to show how *dg2pix* can be utilized to identify temporal changes and states in dynamic graphs.

In summary, the contributions of this work are the following: (1) The novel *dg2pix* visualization technique, a time-scalable visual metaphor to reveal changes and similar temporal states in a dynamic graph; (2) an interpretation strategy of visual patterns that users can examine in *dg2pix*; and (3) an interactive prototype that allows exploring dynamic graphs at multiple scales.

2 Related Work

In the following, we briefly discuss related work from dynamic graph visualizations, the visual analysis of dimensionality reduction methods, and pixel-based visualization techniques.

2.1 Dynamic Graph Visualization

In many application domains, dynamic graph visualization techniques have recently gained more research attention [3]. Such techniques can be classified into two main categories: animation and timeline visualizations [3]. The animation of large-scale dynamic graphs is often regarded as inadequate due to the cognitive efforts to maintain a mental map [37] and trace changes [46]. On the other hand, timeline visualizations map the graph often to a compact representation to reduce the cognitive efforts and to enable the comparison of periods. For instance, the parallel edge splatting technique [12] displays dynamic graphs as a sequence of bipartite graph layouts. However, even in the improved version, that uses the interleaving concept [10], the identification of temporal patterns remains challenging due to the over-drawing problems between the individual graphs. Further, Van den Elzen et al. [47] proposes to extend massive sequence views and suggests different reordering strategies to minimize block overlaps. Nevertheless, identifying temporal patterns in dense and large-scale graphs remains challenging due to the overlapping edges, making it difficult to trace changes in the linear-ordering. An extensive survey of further dynamic graph visualization techniques can be found in the surveys of Kerracher et al. [27], Beck et al. [3], and Nobre et al. [36].

In summary, dynamic graph visualizations such as animations and timeline mappings do not scale to long sequences of large-scale graphs due to limited display space [3]. Therefore, previous visualization approaches apply dimensionality reduction methods to reduce the complexity of the high-dimensional graph data.

2.2 Dimension-Reduced Dynamic Graph Visualization

Visualization approaches based on dimensionality reduction focus on summarizing and abstracting dynamic graphs to highlight temporal and structural changes. For example, Van den Elzen et al. [48] use dimensionality reduction methods (e.g., t-SNE [33]) to reduce the amount of data and provide an overview of high-level temporal states in a dynamic graph. The proposed visual analytics approach, however, depends on the temporal discretization scale and requires feature engineering for embedding the discretized intervals into vectors. Time curves [1] likewise embed the temporal data in a spatial layout to highlight temporal patterns and anomalies. Still, time curves heavily depend on feature engineering of the vectors, the quality of the distance metric, and often produce visual clutter for long sequences due to overlapping issues. For further reading, we refer to the survey Engel et al. [18], Sacha et al. [39], and the work of Vernier et al. [50].

Overall, dimensionality reduction methods reduce the complexity of the dynamic graph data and support the identification of temporal patterns. However, such methods frequently fail to capture structural changes as the methods heavily depend on the designed feature vector (e.g., graph metrics). Furthermore, such methods do not scale to long sequences due to the visual clutter caused by overlaps in the spatial layout. Pixel-based visualization techniques can be utilized to avoid such over-drawing problems.

2.3 Pixel-Based Visualization

Pixel-based visualizations effectively use the whole display space and allow us to present large amounts of data without overlap and clutter [26]. For example, Buchmueller et al. [9] highlighted the usefulness of pixel-based visualizations for the visual summarization of changes in spatio-temporal data. There are, however, only a few pixel-based visualizations for dynamic graphs. For instance, a matrix of pixel-based glyphs can be used to highlight temporal patterns in small social networks [43]. Furthermore, space-filling temporal treemap visualizations [28] can be extended to display medium-sized evolving trees in a pixel-based visualization manner. Such temporal treemaps [28] and other hierarchy based visual metaphors [11] are, however, only able to depict evolving hierarchies. Another pixel-based timeline visualization is GraphFlow [15], which displays graph metrics to highlight structural changes in a dynamic graph. The GraphFlow [15] method, however, only works for small graphs with a limited number of time steps, and the energy-based visualization also heavily depends on the used graph metric (e.g., node degree) that can fail to capture the overall dynamic phenomena.

2.4 Delineation to our Work

In this work, we propose an overlap free multiscale pixel-based visualization that does not require any feature engineering, scales up to long sequences of graphs, and enables us to drill-down into aggregated temporal intervals. We utilize unsupervised analysis methods (graph embeddings) from the field of machine learning to automatically learn and embed graph structures in low-dimensional space without requiring any features engineering [53]. Such graph embedding methods stand in contrast to previous analysis methods (e.g., GraphFlow [15]) that typically used static or dynamic graph metrics (e.g., diameter [7] or change centrality [19]). However, there are currently no visualization techniques that leverage graph embeddings for dynamic graphs even though they have shown to be efficient for various tasks (e.g., in link prediction [21]). Our work was inspired by the stripe-based visualizations of word embeddings that can be used to highlight semantically similar word groups [41]. Contrary to previous approaches, *dg2pix* scales to long and large-scale dynamic graphs, providing an encompassing overview of possible temporal aggregation levels. In this work, we also investigate how the visual patterns in *dg2pix* can be interpreted.

3 Dynamic Graph to Pixel-Based Visualization

dg2pix is a scalable visualization technique to gain an overview of the temporal dimension in long and large-scale dynamic graphs. The approach combines temporal aggregations with dimensionality reduction methods (graph embeddings) at multiple temporal scales to reveal temporal patterns, for instance, reoccurring states with similar graph structures. With *dg2pix*, we show how graph embeddings can be interactively used to surpass state-of-the-art visualizations techniques for dynamic graphs by the amount of displayed information.

The visualization technique consists of three adjustable steps (see Fig. 2). The technique's basic idea is to use graph embeddings combined with a pixel-based visualization to present vast amounts of high-dimensional data to support the exploration and summarization of dynamic graphs [8]. The (2-3) transformation steps (see Fig. 2) are, to the best of our knowledge, not considered in previous literature for dynamic graph visualizations even though graph

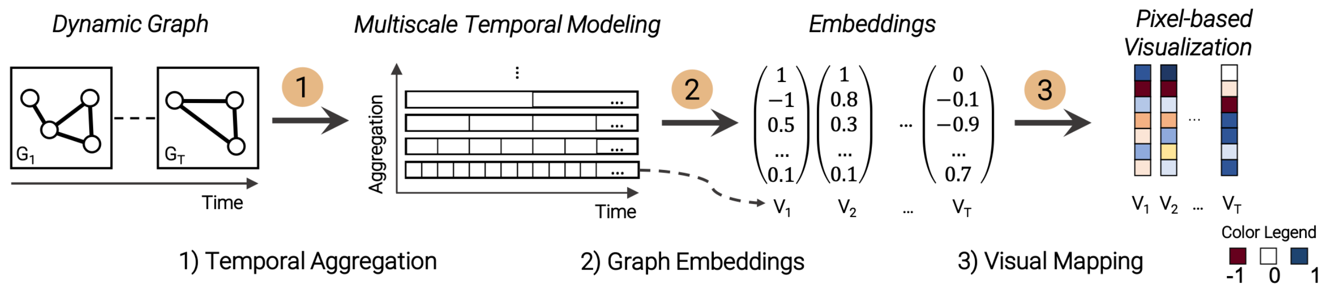


Figure 2: The *dg2pix* enables users to discover similar temporal states. The approach consists of three main adjustable steps (1) temporal aggregations, (2) graph embeddings, and (3) the visual mapping to the pixel-based visualization.

embeddings outperform many state-of-the-art unsupervised learning methods [53], and pixel-based visualizations enable to display large amounts of data overlap and clutter-free [26]. In the following, we explain the three steps of *dg2pix* and present the implemented prototype.

3.1 Multiscale Temporal Modelling

Temporal abstraction methods (e.g., temporal aggregations) are applied to dynamic graphs to reduce the amount of data and summarize the changes over time. Typically, such temporal abstraction methods are based on aggregation, such as the supergraph computation, which summarizes intervals by unifying all nodes and edges of a sequence of graphs [22]. Supergraphs provide an overview of temporal intervals by summarizing a sequence of graphs to only one graph with the cost of discarding temporal information [5]. The computation of supergraphs can be seen as a discretization of the temporal dimension. However, the usefulness of such a temporal discretization depends on many aspects, for instance, graph size, frequency of topological changes, and the temporal aggregation scale [5]. For example, a fine-grained temporal aggregation into supergraphs results in various intervals with little information unable to provide an overview. In contrast, coarse-scale aggregation produces only a few supergraphs, which may contain a high variance, where important intervals may remain hidden. Moreover, finding the optimal fixed interval length for analysis depends on the user task at hand [16].

We compute supergraphs at different temporal aggregation scales to enable users to explore temporal states at multiple temporal granularities interactively. In contrast to previous discretizations of time that use uniform or non-uniform temporal granularities (time-slices) [51], we propose to recursively partition the data using uniform time-slicing methods and compute for each interval a supergraph. For example, the recursive supergraph generation can be done based on the cyclic division of the time, such as the division into a year, months, and days. For domains with no reasonable temporal partitioning, we propose the following default dynamic graph coarsening approach. The default approach slices T time steps of the temporal dimension recursively into intervals of length 2^l with l being the level $l \in 1,...,\lceil log(T) \rceil$. The resulting levels contain at the lowest level one intervals of length one and the highest level $\lceil log(T) \rceil$ a supergraph of all graphs. We compute a supergraph for each of these intervals, which results in $\lceil T/2^{l-1} \rceil$ supergraphs for each the level l. The multiscale temporal modeling computes $\lceil log(T) \rceil$ levels of granularity having overall $(2 \cdot T) + 1$ supergraphs.

In summary, the temporal multiscale modeling essentially recursively coarsens the dynamic graph into supergraphs, which are used in combination with the original evolving graphs in the next step to learn the similarities between graphs in a latent space. Our multiscale temporal modeling was inspired by Elmqvist and Fekete [17] hierarchical aggregation, which enables us to turn visualizations into multiscale (multiresolution) approaches that scale better to large datasets. The multiscale temporal modeling is later used to perform unbalanced drill-down and roll-up operations.

3.2 Graph Embedding

In the second step, we apply dimensionality reduction methods (graph embeddings) to all generated snapshots to learn the similarity between graphs and reduce the high-dimensional data to low-dimensional vectors. We apply graph embeddings (e.g., graph2vec [35]) as they are scalable to large-scale dynamic graphs, outperform state-of-the-art methods in the field of unsupervised learning, do not require feature engineering, and are small enough to fit into main memory for interactive visual analysis [53]. For example, graph2vec outperforms graph kernels, and substructure embedding approaches for classification tasks on large graph datasets [35]. A graph embedding can be seen as a function $f : V \rightarrow \mathbb{R}^d$ that maps a set of vertices (e.g., random walks) to a d dimensional embedding. Typically, the embeddings of the latent space $\mathbb{R}^d$ are used to gain insight into the data and for further standard machine learning tasks. For example, we cluster the embeddings to visualize and gain an overview of similar temporal states. In contrast to previous dynamic graphs visualization approaches (e.g., van Elzen et al. [48]), which depend strongly on the used graph metric, unsupervised graph embeddings do not require any feature engineering, are task agnostic and data-driven. An advantage of graph embeddings is that the methods learn similarity between graphs in the latent space by approximating different graph metrics [6]. Furthermore, we employ graph embeddings instead of node embeddings because graph embeddings only compute one vector for a given time step and therefore scale to large datasets.

We utilize the three recently proposed graph embeddings [38] for our approach: graph2vec [35], GL2Vec [14], and FGSD [49] as the approaches have moderate run-time complexities. We compute embeddings of all $2T+1$ supergraphs of the multiscale temporal modeling step and embed the graphs, as suggested by Bonner et al. [6], into the range of $50-300$ dimensions. Per default, we embed each graph to a vector of 128 features and apply L_2 normalization to the embeddings. The normalization maps the vectors to unit length and enables us to use cosine similarity instead of the dot product as a distance measurement [30]. The vectors are later displayed in the pixel-based visualization as pixel-bars to identify changes in dynamic graphs visually. In Fig. 5, three different graph embeddings of a synthetic dataset are presented, highlighting that the proposed methods capture temporal states in dynamic graphs. In our discussion (see Sec. 6), we elaborate on the input parameters and the scalability of such graph embeddings.

3.3 Pixel-based Visualization

In the last step, we visually encode the embeddings into dense pixel-based visualizations to provide an overview of the temporal dimension and reveal similar graphs. We particularly visualize the embeddings as they are compact encodings of the structural information of each graph. The embeddings are displayed as linearized pixel-bars that are basically grid-based columns in which each rectangle (pixel) is a feature of the embedded vector. The technique colors each pixel by the feature's value by using a diverging color scheme from ColorBrewer [23]. We utilize a global segmented color scheme with two distinct values to support the comparison task [44]. We sequentially order each displayed pixel-bar (graph embedding) by time, creating a dense pixel-based visualization.

The y-axis ordering of the colored pixel-bars is per default based on the linear order of the vector. The challenge of finding a useful linear order to highlight particular patterns visually can be mapped to the linearization problem [4]. We are utilizing different reordering algorithms to improve the global ordering of the embeddings to emphasize different patterns along both axes of the pixel-based visualization. For example, we apply clustering algorithms to all displayed data features to group and arrange similar features over time. We discuss different reordering strategies in Sec. 4.3.

We utilize the computed supergraphs of the multiscale temporal modeling to present the data at multiple user-defined levels of temporal aggregation (see Fig. 4). Such a multiscale (multiresolution) visualization helps to set detailed abstraction levels into the overall temporal context [17]. For example, the visualization technique presents 1000 supergraphs as pixel-bars instead of several thousand individual graphs and enables users to drill-down into intervals. We limit the number of depicted grid-based columns to the available horizontal pixels of the screen space to address our approach's visual scalability, which means that the minimum width of a pixel-bar is precisely one pixel. If a user drills down and reaches the limit of screen space pixel, he has to coarsen temporal intervals to reduce the number of overall displayed pixel-bars. Next, we describe our implemented prototype and available interaction methods.

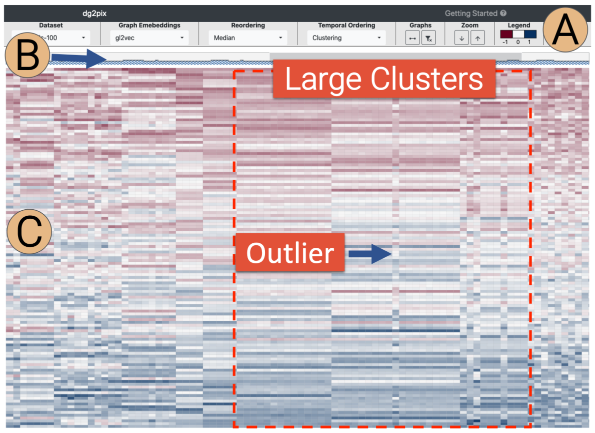
3.4 Prototype

The *dg2pix* prototype implementation [1] enables us to explore the temporal changes of large scale dynamic graphs. In the following, we briefly introduce the two main linked views of the prototype.

The ***dg2pix* view** (see Fig. 3) consists of a *toolbar* (A), a *zoom context bar* (B), and the *pixel-based visualization* (C). The *toolbar* allows selecting and presenting various graph embeddings for particular datasets, including choosing different training epochs and applying automated analysis methods. For example, the x-axis can be reordered based on a clustering of the graph embeddings (see Sec. 4.3). Furthermore, the toolbar enables us to change the temporal granularity of intervals (drill-down and roll-up) and display selected graph embeddings in the graph view. The *zoom context bar* presents additional information for the vertical and horizontal temporal navigation and provides an overview of the displayed temporal interval and granularities. The view (see also Fig. 4) displays for each pixel-bar the corresponding temporal granularity as a zoom bar (rectangle). The height is mapped to the zoom level, meaning the zoom bars of low levels of temporal granularity are rather small. The zoom bars are always ordered by time and enable us to relate the potentially reordered pixel-bars to their overall temporal context via brushing and linking. The zoom context bar also allows for selecting and filtering periods of the pixel-based visualization, allowing navigating horizontally along the temporal dimension.

The *pixel-based visualization* displays per default the medium zoom level of the graph embeddings ordered by time. The view allows us to select individual and multiple pixel-bars and adapt the temporal granularity by drilling-down a lower temporal granularity or coarsening the temporal dimension (roll-up). The view is also directly linked to the zoom context bar, enabling us to keep an overview and relate the pixel-bars to the temporal dimension. The x- and y-axis of the pixel-based visualization can also be reordered using different reordering strategies to highlight clusters and similarities between the embeddings (see Sec. 4.3). Furthermore, multiple pixel-bars can be selected to display the underlying supergraphs and graphs in the second main view.

[1] https://github.com/eren-ck/dg2pix



Dataset
-100
Graph Emebeddings
gl2vec
Reordering
Median
Temporal Ordering
Clustering
Graphs
Zoom
Legend
-1 0 1

Figure 3: The *dg2pix* component consists of three views a toolbar (A), the zoom context bar (B), and the pixel-based visualization (C). In (C) the embeddings (GL2Vec [14]) of a synthetic dynamic graph (SBM [25]) with reoccurring states are depicted. The displayed embeddings of the x-axis are clustered (HDBSCAN [13]), and y-axis ordering is based on the median of each vector attribute. The reordering and clustering of the synthetically created reoccurring states highlight large clusters of similar graphs and outliers in the temporal data.

The **graph view** allows us to display the underlying graph data of the selected pixel-bars as a supergraph to highlight and compare the intersections and disjoint nodes and edges between the graphs in the temporal data. The supergraph nodes and edges are colored using two graph set operations on all selected time steps to highlight similarities and differences. The applied set operations are intersection (orange) and disjoint (blue) on all nodes and edges of the selected time steps. The main goal of this comparison using set operations is to investigate the changes in the temporal graph, which helps to identify and interpret which graph structures were preserved in the latent space. The view uses per default for all time steps one precomputed graph layout (Fruchterman-Reingold [20]) by computing a supergraph for the whole dynamic graph. We facilitate one global layout for the whole dynamic graph to preserve the user's mental map [2]. The graph view can also be explored via semantic zooming to explore particular graph structures, such as specific node and link attributes.

4 Visual Interpretation of dg2pix

dg2pix provides a scalable overview of the temporal dimension to emphasize underlying changes in dynamic graph data. The main idea of the approach is to learn and display low-dimensional embeddings of graphs that capture the similarity between graphs in a latent space. However, the interpretation of such embeddings in the latent space remains challenging as the meaning of particular numeric values cannot be directly mapped to topological features of the graph. For example, the specific meaning of a dimension value of 0.3 of an embedding with 128 dimensions remains unanswered. Consequently, the abstractness of what low and high values of each dimension encode poses a challenge to understand and map the patterns in *dg2pix* to topological changes in the evolving graph. In previous work, typically, 2D visualizations are used to interpret and understand such latent space [32]. For instance, the Embedding Projector [42] by Google Brain uses projections (e.g., t-SNE [33]) to present word embeddings as 2D and 3D scatterplots. However, such simple 2D visualizations discard latent space information as the d-dimensional embeddings are again reduced into a 2D embeddings for the visual representation. The following section describes the underlying challenges of visualizing latent spaces, the interpretation of visual patterns, and different reordering strategies to highlight temporal changes.

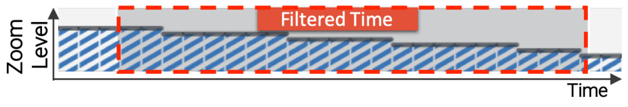


Figure 4: The zoom context bar enables us to investigate the zoom level for an individual and multiple pixel bars. Further, it allows filtering time intervals for vertical and horizontal navigation.

4.1 Latent Space Visualizations

Recently, the visual analysis of latent spaces (embedding spaces) has gained research interest [32]. For example, ad-hoc dimensionality reduction methods (e.g., PCA, t-SNE [33], or UMAP [34]) are often applied to display neighbors in the latent space in 2D space. The latent space representation central goal is to provide more insight into the underlying embedded data and enable the qualitative interpretation of the learned embeddings [32]. Heimler and Gleicher [24], for instance, describe tasks for word embeddings and display words in a matrix-based view to highlight co-occurrences between words. Further, Liu et al. [32] describe a set of tasks for exploring latent spaces and present a cartography system to visually investigate relationships between data points and compare attributes of vectors (e.g., word embeddings). The visual analysis of latent spaces currently remains the primary method to investigate and interpret graph embeddings. There has been little theoretical work to prove that such embeddings approximate and learn different graph metrics [6]. For example, EmbeddingVis [31] enables the comparison of different latent spaces of node embeddings to investigate which node metrics are preserved by applying regression.

In contrast to all previous approaches, our primary goal is to generate a visual summary of the temporal dimension that helps to understand and highlight temporal states in the evolving data. We display the embeddings with all their dimensions to visually compare similarities and apply reordering strategies to present changes in the latent space. Our approach also allows us to present the underlying graphs in combination using graph set operations (e.g., union or intersection) to help interpret and compare the latent space with the original evolving graph data. Next, we elaborate on how *dg2pix* can be interpreted, and automatic approaches can be used to find similar temporal states.

4.2 Interpretation of Visual Patterns

Graph embeddings are machine learning models that produce abstract low-dimensional vector representations for graphs that are difficult to interpret, as the individual values of the dimension themself have no exact interpretation [40].

Challenges The reasons for interpretation challenges arise from the stochastic algorithms (e.g., graph2vec [35]), which utilize non-transparent neural networks with hyperparameters [40]. Further, the embeddings can be changed with unitary rotation, which completely transforms each dimension's values while preserving the latent spaces distances. Therefore, the complexity of interpreting graph embedding dimensions can be compared to the efforts to understand activations in neural networks for image classification [40]. Nevertheless, recent experiments [6, 40] indicate that graph embedding methods learn to approximate various topological features of graphs. Therefore, we utilize and visualize graph embeddings to highlight changes in dynamic graphs as the methods have shown to be effective feature spaces for various graph mining tasks, such as classification of graphs [14, 45, 49].

Interpretation The pixel-based visualization enables us to perceive similarities and differences between embeddings to provide an overview of the dynamic graph. Generally, the visualization of embeddings can reveal relationships in the latent space, as shown by Shin et al. [41] for the comparison of semantically similar word embeddings. The graph embeddings can only be interpreted in relation to other embeddings by investigating the pairwise similarity between embeddings. More specifically, if two subsequent graph embeddings in the dynamic graph are, to some extent, similar to each other, then the original graphs are also similar to one another. Also, vice versa, if two successive embeddings are different, then the two embedded graphs are dissimilar to some extent. Therefore, we can use the embeddings to examine and highlight changes and temporal states in a dynamic graph even though we cannot interpret the individual values of particular dimensions.

Visual Comparison of Embeddings Consequently, the human-centric visual analysis of temporal states (e.g., reoccurring graphs) can be mapped to distinguishing similar pixel-bars in the *dg2pix* . For instance, Fig. 3 displays a large block of similar pixel-bars with an apparent outlier in-between. The visual analysis of pairwise similarities between pixel-bars enables identifying temporal changes and states in the underlying dynamic graph. However, the cognitive efforts to compare multiple pixel-bars are high since the user has to simultaneously relate numerous dimensions of different embeddings. The pairwise similarities between multiple embeddings can also be computed using the cosine similarity. We, therefore, propose to use automatic methods to sort and cluster similar rows and columns in the pixel-based visualization to enable the identification of temporal states (e.g., outliers) in the dynamic graph.

Explainability We also compare the underlying graph structures of embeddings in the graph view against each other, intending to generate new insight into the latent space. For instance, displaying the graph data helps to explain potential reasons and impacts of graph features on particular values for individual dimensions. Overall, both the pixel-based visualization and the graph view can help to understand and explain the semantic meaning of high and low values of particular dimensions to gain new insight into graph embeddings, which are currently black-box models [40].

4.3 Reordering Strategies

dg2pix was designed to scale to large-scale dynamic graphs and provide a visual summary of the temporal data. However, temporal states can remain hidden and difficult to identify due to the sheer amount of visualized data, for instance, if single reoccurring pixel columns correlate with other prominent states. Applying different reordering strategies to the embedding can reveal such otherwise hidden temporal states. For example, clustering reordering the displayed pixel columns (x-axis) will highlight similar graph structures. Therefore, we provide users with the option to apply reordering strategies to reveal similar patterns along both axes.

We provide global reordering strategies for the dimensions of the embeddings (y-axis) and the temporal dimension (x-axis). In general, identifying an optimal ordering for our pixel-based visualization is known to be NP-Hard since the issue can be mapped to the problem of reordering (linearization) of rows and columns in matrices [4]. For the reordering of matrices, various reordering strategies (layouts) have been proposed to highlight different patterns (e.g., block patterns [4]). We provide for the reordering of the embedding dimensions (y-axis) several heuristics based on a statistical metric of each row. For example, before the L_2 normalization, the y-axis can be sorted by the median value for each dimension of the displayed embeddings to highlight block and band patterns [4]. Furthermore, the prototype allows us to reorder the dimensions (rows) of the pixel-based visualization using the mean, minimum, maximum, variance, and standard deviation of the depicted rows.

We also provide two reordering strategies for the temporal dimension (x-axis) to identify similar temporal states by computing clusters and reordering based on the distances to one particular column (similarity search). The clustering uses HDBSCAN [13] for the displayed embeddings facilitating the cosine-similarity as a distance measurement. We employ HDBSCAN [13] as the approach aims to find the result with the best stability over different epsilons parameters and accordingly detect clusters with varying densities. The clustering results are displayed by grouping and highlighting

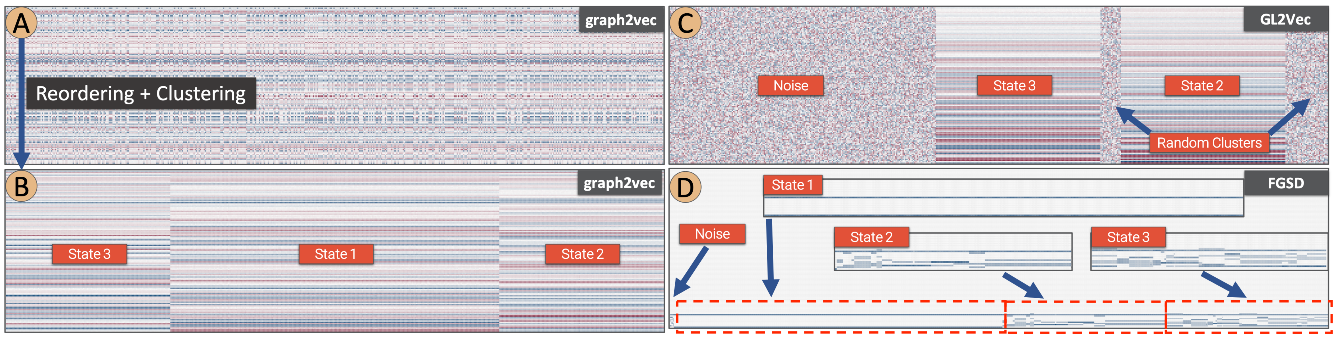


Figure 5: The synthetic dynamic graph described in the use cases (see Sec. 5.1) displays three different graph embeddings with a ground truth of three temporal states. (A) presents the dynamic graph using graph2vec [35], and (B) shows the same data with the three temporal states. In (B-D), the same reordering strategies were applied to highlight the temporal states. The two other graph embeddings, (C) GL2Vec [14] and (D) FGSD [49], are partially able to learn and highlight the three temporal states in the synthetic dynamic graph.

the pixel-bars according to their clusters using a grey bounding box. For instance, the clusters are reordered using the median time of all embeddings, and the underlying embeddings of a cluster are again sorted by time. Second, we enable to reorder the y-axis based on the distance to a particular embedding. The resorting places an embedding to the first position and afterward ranks the presented embeddings by the distance to the selected embedding (similarity search). This reordering enables us to compare one particular embedding in time with all other graph embeddings in detail.

Overall, using such reordering strategies for both axes can help users understand how the ordering influences the visual patterns, can group, and rank similar temporal states to explore the latent space in more detail.

5 Use Cases

In the following section, we apply *dg2pix* to synthetic and a real-world dynamic graph to demonstrate how the approach can be used to gain an overview and provide insight of the temporal changes and reoccurring states in evolving graphs.

5.1 Synthetic Dynamic Graphs

We generated synthetic dynamic graphs, with known ground truths, to show the applicability and the usefulness of *dg2pix* . For example, we created different datasets with the Stochastic Block Model (SBM) [25] with a fixed amount of nodes for each time step, a varying number of edges, and multiple temporal states (see Fig. 1). We elaborate on the results of one dynamic graph to show how the approach can be used to identify states in large-scale graphs.

The synthetic dynamic graph consists of 1000 time steps, 1000 nodes, more than 30 million edges, and three reoccurring temporal states. We facilitated the SBM to create three states with different numbers of clusters (blocks), a slightly varying number of nodes (up to 50) per cluster, and minor edge density changes (internal and external). The dynamic graph consists of randomly shuffled data of 500-time steps with two clusters of nodes, 250-time steps with three clusters, and 250-time steps with four clusters. The dynamic graph was embedded with three different graph embeddings with the following parameters:

- **graph2vec** [35]: 1000 epochs, 0.02 learning rate, 2 Weisfeiler-Lehman iterations, and 128 dimensions.
- **GL2Vec** [14]: 1000 epochs, 0.02 learning rate, and 128 dimensions.
- **FGSD** [49]: 128 number of histogram bins with a the histogram range of 20.

The Fig. 5 (A-D) shows the resulting *dg2pix* of the synthetic dynamic graph. In Fig. 5 (A), the randomly shuffled data is displayed using the graph2vec [35] embeddings, and in (B) the same pixel-bars are presented after the application of reordering strategies. We reordered the embeddings (x-axis) based on the clustering of the embeddings (HDBSCAN [13]), and the rows were globally sorted based on the standard deviation of each row (ascending). The reordering strategies help to identify temporal and reoccurring states (e.g., clusters) by grouping similar and dissimilar pixel-bars and their respective rows together. For example, sorting the rows by the standard deviation of each row allows users to compare and identify the embedding dimensions that primarily distinguish temporal states. In Fig. 5 (B), the three temporal states are visible, which can be verified by displaying the underlying graph structures in the graph view. Accordingly, graph2vec has managed to learn the temporal states encoded in the underlying ground truth.

In contrast, GL2Vec [14] was not able to distinguish the three temporal states (see Fig. 5 (C)). The same reordering strategies result in only two visible temporal states. The GL2Vec model learned to distinguish the states with the three and four clusters, however, the model was not able to distinguish the larger group of two clusters (500-time steps) in the latent space. The clustering grouped the first temporal state as the visible block of noise and identified two similar states in the ground truth as two different clusters. The GL2Vec model potentially requires a different learning rate or more epochs to distinguish the third state in the latent space.

In Fig. 5 (D), the FGSD [49] is displayed which approximately learns the three temporal states. Compared to the first two methods, the FGSD model embeds the dimensions only to a positive range (blue color), and only seven dimensions of the embeddings contain values. The method is almost able to distinguish all three clusters except for a little bit of noise, which can be verified by visualizing the graphs in the graph view. In contrast to the other graph embeddings, the FGSD model results in considerable white space that can be removed by deleting rows that do not contain any values.

In addition to the different synthetic graphs with known ground truth, we also created random dynamic graphs with different graph generators to confirm that the visible patterns are not arbitrarily learned in the latent space during the training process. For instance, Fig. 6 shows a dynamic graph with 1000 randomly generated connected Watts–Strogatz small-world graph [52] with 2000 nodes (between 5-50 nearest neighbors), and < 0.1 edge probability edges for each time step. The same reordering strategies, as in Fig. 5, were applied, and the resulting *dg2pix* shows the graph2vec (1000 epochs) embedding, which does not contain any visible patterns as the model was not able to learn the similarities between the random graphs in latent space.

graph2vec
Random Graphs

Figure 6: The synthetic random dynamic graph contains no known ground truth, and the proposed approach produces no visible patterns.

5.2 Evolving Social Network

Next, *dg2pix* is applied to a real-world, large-scale social networks. We describe the temporal visual analysis of the website Reddit [29] to discover structural and temporal changes as well as reoccurring states between social network communities (subreddits) during the 2016 US presidential elections. In the following, we describe the analyzed dataset, highlight the main task and challenges for the analysis of such data, and how *dg2pix* can be used to provide an overview of the temporal changes.

Reddit Data Reddit is a social news aggregation website with approximately 440 million users as of 2020. The website is made up of subreddits in which users post content (e.g., images or links to news sites) and upvote posts based on a voting based system to rank interesting content for each subreddit. The dataset [29] is a dynamic hyperlink graph and consists of subreddits (nodes), and time-stamped hyperlinks (edges). The analyzed data contains hyperlink graphs grouped by hours from the 1st January 2016 to 30th November 2016 in which the election campaign for the 2016 presidential election took place. The dynamic graph consists of 7974 graphs, 18546 nodes (subreddits), and 88328 edges (hyperlinks) between the subreddits with either positive or negative sentiment. We computed the following three graph embeddings graph2vec [35], GL2Vec [14], and FGSD [49] with the same input parameters as described in Sec. 5.1. We verified the resulting insight by comparing the identified changes and states of the underlying evolving hyperlink graphs to the real historic news coverage of the presidential elections.

Tasks and Challenges The visual analysis of a social network data aims to provide an overview of structural changes over time, temporal states (e.g., reoccurring graph structures), and outlier graphs in the evolving data (e.g., political scandals). However, gaining an overview of large-scale social media data is challenging as it requires to visualize structural as well as temporal changes simultaneously and to identify suitable temporal analysis scales for changes and states of varying temporal length. Furthermore, the size and complexity of social networks pose another challenge to visualize the evolving data since there is a trade-off between the visualization of the detailed graph structure for each time step and presenting the overall evolving graph properties. For instance, animations display each graph of the data in detail, however, animations are considered to be unsuited to provide an overview of long periods due to cognitive efforts to keep track of changes [46]. In contrast to previous approaches, we model and embed dynamic graphs at multiple temporal scales to enable the multiscale temporal analysis of long as well as large-scale dynamic graphs.

2016 US Presidential Election We begin by investigating the week before and the week during the 2016 US presidential elections (8th November 2016) to identify graphs with political subreddits in the temporal data. Per default, the prototype displays 400 pixel-bars of the middle level of temporal granularity using the graph2vec [35] embeddings. First, we use the multiscale temporal modeling to concentrate on the election weeks in November 2016. We aggregate the pixel-bars before October into aggregated supergraphs (roll-up), and further split (drill-down) the election weeks into the lowest temporal granularity of one hour. We display different graph embeddings to examine the resulting pixel-bars during the election week period visually. We decide to use the GL2Vec [14] embeddings, as there are some noticeable similar pixel-bars in the *dg2pix* (see Fig. 7 (A)) in which the x-axis is sorted by time. Next, we apply the implemented reordering strategies to group and highlight similar pixels-bars. The median of each row reorders the y-axis, and we cluster and reorder the embeddings of the x-axis (see Fig. 7 (B)). The first visibly large group of graph embeddings is classified as noise as the embeddings seem to have distinct values in the latent space. The next groups are clustered together and also have visually similar looking embeddings. We investigated the graphs in groups and between groups by displaying and comparing them in the graph view. Thereby, we interpreted and tried to link the embedding characteristics to evolving graph structures. For example, we noticed that the first group consists of many computer games subreddits (e.g., *pokemongo*) and that the following group contains various political subreddits (e.g., *the_donald*, *AskTrumpSupporters*, or *politics*). We were also able to identify graph structures related to specific time aspects. For example, the last group (AM) consists of hyperlinks posted only in the morning (between 8-11 am). These graphs posted in the morning have specific characteristics (e.g., fewer subreddits) that have been learned by the graph embedding.

Searching for Political Events Next, we search for political events during the 2016 presidential election to identify graph structures with hyperlinks between political subreddits. First, we change the temporal granularity of all embeddings to the duration of 8 hours, which results in approximately 1000 pixel-bars. We select the election night of the 8th of November (6 pm - 12 am). We assume that political subreddits, which posted hyperlinks to other subreddits during the election night, were also active during the election campaign. Afterward, we use the ranking functionality to search in all three graph embeddings for similar embeddings, and we examine the top results. The top five-nearest neighbors in the three graph embeddings reveal different political events. For instance, graph2vec and GL2Vec return the 1st February can be directly linked to Iowa's democratic and republican caucus. Other graphs resulting from the similarity search can be related to the democratic nomination of Hillary Clinton (28th July) and Mike Pence being announced as the running mate of Donald Trump (15th July). Furthermore, FGSD [49] ranks the 23rd July high, which can be associated with the Wikileaks email release that revealed a bias of the Democratic Party against Bernie Sanders. The publications of Wikileaks are particularly visible in the graph view, as some political subreddits are linked (e.g., *SandersForPresident*, *politics*, *political_revolution*). Overall, the use cases describe how *dg2pix* enables identifying temporal changes and states (e.g., political events) and relating the latent space to structural changes in the underlying graph.

6 Discussion

The cornerstone of *dg2pix* is the visual analysis of embedded graphs as pixel-based visualization to identify temporal states. The visualization technique consists of three steps: (1) the multiscale temporal modeling, (2) graph embeddings, (3) the visual analysis of the pixel-based visualization. In the following, we discuss the limitations of *dg2pix* and potential future research directions.

Parameters The (1-2) step has multiple input parameters that profoundly influence the perceived patterns in the pixel-based visualization, such as the latent space size, number of epochs, or the random initialization of the neural network. Currently, the parameter choices are set by the user as they depend on many factors, for example, the temporal aggregation depends on the discretization scale of the application domain. We consider the usage of various parameters as an advantage of our approach and a possibility for future work to investigate which parameter combinations (e.g., different graph embeddings) can capture distinct temporal changes, such as reoccurring motifs or outlier graphs.

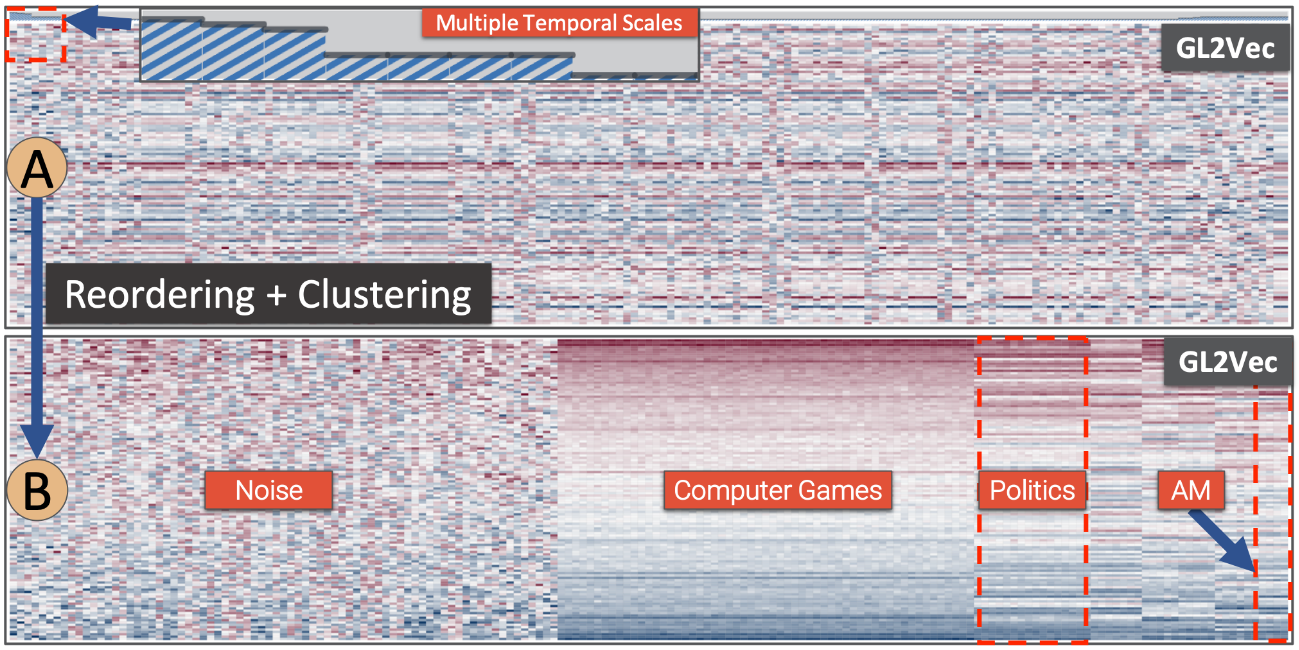


Figure 7: The Reddit data described in the use cases (see Sec. 5.2) presents the election week of the 2016 US presidential election using GL2Vec [14] embeddings. The multiscale temporal modeling was used to display dill-into the election week and aggregate other intervals into supergraphs. (A) displays the evolving social networks sorted by time, and (B) shows the same data after applying reordering strategies to emphasize temporal states. We linked the clusters of embeddings to hyperlinks between different communities of subreddits, for example, computer games related topics, political topologies, or morning graphs structures (AM).

Interpretablity The interpretation of the resulting perceivable changes remains challenging due to multiple reasons (see Sec. 4), which affects the usability of the approach as the visual encoding is challenging to read. We consider the interpretation limitation as minor as our approach focuses mainly on highlighting temporal changes. However, we aim to support the latent space's visual analysis by presenting the underlying embedded graph structures, enabling us to generate new insight into the evolving data and lead to new interpretations. We also offer reordering strategies to examine and interpret neighborhoods and clusters of embeddings in the latent space. Nevertheless, the extension with further contextual features (e.g., evolving graph metrics) is essential to allow a detailed interpretation and guide users towards interesting patterns.

Graph Embeddings We apply unsupervised graph embeddings to reduce the dimensionality of long sequences of dynamic graphs and automatically learn similarities between large-scale graphs. In contrast to topological graph metrics (e.g., density), such unsupervised graph embeddings scale to large graphs, do not require any feature engineering, and are domain as well as task agnostic. The main limitation of such embeddings is that it remains unclear how many embedding dimensions are required to capture specific structural changes [40]. We plan to investigate the required number of dimensions for synthetic temporal patterns and how different input parameters and noise influences the resulting embeddings.

Scalability For the computational scalability, we consider the graph size ($|V|$ nodes and $|E|$ edges) and the number of time steps T. The (1) step computes supergraphs at multiple levels and requires $O(log(T) \cdot (|V|+|E|))$ memory and time complexity. The computation of the supergraphs can be parallelized to increase the approach's scalability to long sequences of graphs. For further reading of runtime complexities of graph embeddings, we refer to the survey of Goyal and Ferrara [21], which emphasizes that recent graph embeddings run in $O(|E|)$. Therefore, the overall runtime complexity of the approach is $O(log(T) \cdot (|V|+|E|))$. We suggest precomputing the embeddings for large scale dynamic graphs on GPU servers, due to the time and memory complexities. Once the embeddings have been calculated, they are small enough to fit into the main memory. Second, the computational efforts affect the interactive visual analysis of the *dg2pix*. For example, the reordering strategy by clustering scales linearly to the displayed time steps and embedding dimensions. Also, the visualization of large scale graphs for the comparison and interpretation in the graph view does not scale to large-scale graphs as the size impairs the node-link diagram's readability. A possible solution for this issue is to cluster the underlying large-scale graphs and display the identified clusters. However, such a clustering makes it challenging to compare graphs as nodes and edges are abstracted into meta-nodes. Therefore, we plan to examine how different graph embeddings, combined with evolving graph metrics, can be used to compare large-scale graphs.

7 Conclusion

We presented *dg2pix*, a visualization technique to provide an overview of temporal changes in long and large-scale dynamic graphs. The novel representation consists of the multiscale temporal modeling, unsupervised graph embeddings, and a dense pixel-based visualization to explore the embeddings at different temporal scales. The main idea is to visually analyze the latent space to identify temporal changes in the dynamic graph. The implemented prototype and the use cases show how *dg2pix* can be used to provide insight into evolving graphs and highlight the applicability of the approach to synthetic and real-world dynamic graph data. Overall, the *dg2pix* is a promising new research direction for dynamic graphs and can be generalized for the visual analysis of unsupervised embedding methods and latent spaces.

Acknowledgments

This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2117 - 422037984 and the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892.

REFERENCES

[1] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE transactions on visualization and computer graphics*, 22:559–568, 2015.

[2] F. Beck, M. Burch, and S. Diehl. Towards an aesthetic dimensions framework for dynamic graph visualisations. In *2009 13th International Conference Information Visualisation*, pp. 592–597. IEEE, 2009.

[3] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. In *Computer Graphics Forum*, vol. 36, pp. 133–159. Wiley Online Library, 2017.

[4] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. In *Computer Graphics Forum*, vol. 35, pp. 693–716. Wiley Online Library, 2016.

[5] S. Bender-deMoll and D. A. McFarland. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2):1–38, 2006.

[6] S. Bonner, I. Kureshi, J. Brennan, G. Theodoropoulos, A. S. McGough, and B. Obara. Exploring the semantic content of unsupervised graph embeddings: An empirical study. *Data Science and Engineering*, 4(3):269–289, 2019.

[7] U. Brandes and D. Wagner. Analysis and visualization of social networks. In *Graph drawing software*, pp. 321–340. Springer, 2004.

[8] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics*, 19(12):2376–2385, 2013.

[9] J. Buchmüller, D. Jäckle, E. Cakmak, U. Brandes, and D. A. Keim. Motionrugs: Visualizing collective trends in space and time. *IEEE transactions on visualization and computer graphics*, 25(1):76–86, 2018.

[10] M. Burch, M. Hlawatsch, and D. Weiskopf. Visualizing a sequence of a thousand graphs (or even more). In *Computer Graphics Forum*, vol. 36, pp. 261–271. Wiley Online Library, 2017.

[11] M. Burch, M. Raschke, A. Zeyfang, and D. Weiskopf. A scalable visualization for dynamic data in software system hierarchies. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*, pp. 85–93. IEEE, 2017.

[12] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011.

[13] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172. Springer, 2013.

[14] H. Chen and H. Koga. Gl2vec: Graph embedding enriched by line graphs with edge features. In *International Conference on Neural Information Processing*, pp. 3–14. Springer, 2019.

[15] W. Cui, X. Wang, S. Liu, N. H. Riche, T. M. Madhyastha, K. L. Ma, and B. Guo. Let it flow: a static method for exploring dynamic graphs. In *2014 IEEE Pacific Visualization Symposium*, pp. 121–128. IEEE, 2014.

[16] P. Devineni, E. E. Papalexakis, D. Koutra, A. S. Doğruöz, and M. Faloutsos. One size does not fit all: Profiling personalized time-evolving user behaviors. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pp. 331–340, 2017.

[17] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010.

[18] D. Engel, L. Hüttenberger, and B. Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering-Proceedings of IRTG 1131 Workshop 2011*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[19] P. Federico, J. Pfeffer, W. Aigner, S. Miksch, and L. Zenk. Visual analysis of dynamic networks using change centrality. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pp. 179–183. IEEE Computer Society, 2012.

[20] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[21] P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[22] S. Hadlak, H. Schumann, C. H. Cap, and T. Wollenberg. Supporting the visual analysis of dynamic networks by clustering associated temporal attributes. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2267–2276, 2013.

[23] M. Harrower and C. A. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[24] F. Heimerl and M. Gleicher. Interactive analysis of word vector embeddings. In *Computer Graphics Forum*, vol. 37, pp. 253–265. Wiley Online Library, 2018.

[25] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[26] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on visualization and computer graphics*, 6(1):59–78, 2000.

[27] N. Kerracher, J. Kennedy, and K. Chalmers. A task taxonomy for temporal graph visualisation. *IEEE transactions on visualization and computer graphics*, 21(10):1160–1172, 2015.

[28] W. Köpp and T. Weinkauf. Temporal treemaps: Static visualization of evolving trees. *IEEE transactions on visualization and computer graphics*, 25(1):534–543, 2018.

[29] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 933–943. International World Wide Web Conferences Steering Committee, 2018.

[30] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[31] Q. Li, K. S. Njotoprawiro, H. Haleem, Q. Chen, C. Yi, and X. Ma. Embeddingvis: A visual analytics approach to comparative network embedding inspection. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 48–59. IEEE, 2018.

[32] Y. Liu, E. Jun, Q. Li, and J. Heer. Latent space cartography: Visual analysis of vector space embeddings. In *Computer Graphics Forum*, vol. 38, pp. 67–78. Wiley Online Library, 2019.

[33] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[34] L. McInnes, J. Healy, N. Saul, and L. Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[35] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[36] C. Nobre, M. Meyer, M. Streit, and A. Lex. The state of the art in visualizing multivariate networks. In *Computer Graphics Forum*, vol. 38, pp. 807–832. Wiley Online Library, 2019.

[37] H. C. Purchase, E. Hoggan, and C. Görg. How important is the "mental map"?–an empirical investigation of a dynamic graph layout algorithm. In *International Symposium on Graph Drawing*, pp. 184–195. Springer, 2006.

[38] B. Rozemberczki, O. Kiss, and R. Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, 2020.

[39] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE transactions on visualization and computer graphics*, 23(1):241–250, 2016.

[40] F. Salehi Rizi and M. Granitzer. Properties of vector embeddings in social networks. *Algorithms*, 10(4):109, 2017.

[41] J. Shin, A. Madotto, and P. Fung. Interpreting word embeddings with eigenvector analysis. In *32nd Conference on Neural Information Processing Systems (NIPS 2018), IRASL workshop*, 2018.

[42] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

[43] K. Stein, R. Wegener, and C. Schlieder. Pixel-oriented visualization of change in social networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pp. 233–240. IEEE, 2010.

[44] C. Tominski, G. Fuchs, and H. Schumann. Task-driven color coding. In *2008 12th International Conference Information Visualisation*, pp. 373–380. IEEE, 2008.

[45] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2347–2356, 2018.

[46] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: can it facilitate? *International journal of human-computer studies*, 57(4):247–262, 2002.

[47] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Dynamic network visualization with extended massive sequence views. *IEEE transactions on visualization and computer graphics*, 20(8):1087–1099, 2013.

[48] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE transactions on visualization and computer graphics*, 22(1):1–10, 2016.

[49] S. Verma and Z.-L. Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pp. 88–98, 2017.

[50] E. F. Vernier, R. Garcia, I. P. d. Silva, J. L. D. Comba, and A. C. Telea. Quantitative Evaluation of Time-Dependent Multidimensional Projection Techniques. *Computer Graphics Forum*, 2020. doi: 10.1111/cgf.13977

[51] Y. Wang, D. Archambault, H. Haleem, T. Moeller, Y. Wu, and H. Qu. Nonuniform timeslicing of dynamic graphs based on visual complexity. In *2019 IEEE Visualization Conference (VIS)*, pp. 1–5. IEEE, 2019.

[52] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[53] D. Zhang, J. Yin, X. Zhu, and C. Zhang. Network representation learning: A survey. *IEEE transactions on Big Data*, 2018.