Representation Integrity in Temporal Graph Learning Methods

Elahe Kooshafar, School of Computer Science

McGill University, Montreal

July, 2025

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Computer Science

Abstract

Real-world systems ranging from airline routes to cryptocurrency transfers are naturally modelled as *dynamic graphs* whose topology changes over time. Conventional benchmarks judge dynamic–graph learners by a handful of task-specific scores, yet seldom ask whether the embeddings themselves remain a truthful, interpretable reflection of the evolving network. We formalize this requirement as **representation integrity** and derive a family of indexes that measure how closely embedding changes follow graph changes. Three synthetic scenarios—*Gradual Merge*, *Abrupt Move*, and *Periodic Re–wiring*—are used to screen forty–two candidate indexes. Based on which we recommend one index that passes all of our theoretical and empirical tests. In particular, this validated metric consistently ranks the provably stable UASE and IPP models highest. We then use this index to do a comparative study on representation integrity of common dynamic graph learning models. This study exposes the scenario–specific strengths of neural methods, and shows a strong positive rank correlation with one-step link-prediction AUC. The proposed integrity framework, therefore, offers a task-agnostic and interpretable evaluation tool for dynamic-graph representation quality, providing more explicit guidance for model selection and future architecture design.

Abrégé

Les systèmes réels, allant des itinéraires aériens aux transferts de cryptomonnaies, sont naturellement modélisés comme des graphes dynamiques dont la topologie évolue au fil du temps. Les benchmarks conventionnels évaluent les apprenants de graphes dynamiques selon quelques scores spécifiques à chaque tâche, mais se demandent rarement si les représentations elles-mêmes restent un reflet fidèle et interprétable de l'évolution du réseau. Nous formalisons cette exigence par l'intégrité de la représentation et dérivons une famille d'indices qui mesurent la proximité entre les variations de représentation et les variations du graphe. Trois scénarios synthétiques : Fusion graduelle, Déplacement brusque et Recâblage périodique, sont utilisés pour sélectionner quarante-deux indices candidats. Sur cette base, nous recommandons un indice qui réussit tous nos tests théoriques et empiriques. En particulier, cette mesure validée classe systématiquement les modèles UASE et IPP dont la stabilité est prouvée en tête. Nous utilisons ensuite cet indice pour réaliser une étude comparative de l'intégrité de la représentation des modèles courants d'apprentissage de graphes dynamiques. Cette étude expose les atouts des méthodes neuronales, spécifiques à chaque scénario, et montre une forte corrélation positive des rangs avec l'AUC de prédiction de lien en une étape. Le cadre d'intégrité proposé offre donc un outil d'évaluation de la qualité de la représentation des graphes dynamiques, indépendant des tâches et interprétable, fournissant des indications plus explicites pour la sélection des modèles et la conception des architectures futures.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Professor Reihaneh Rabbany, for her time and efforts in guiding this work, reviewing this thesis, and supporting my research throughout my graduate studies.

I am also deeply thankful to my friends and family for their encouragement and support through every challenge. This research would not have been possible without you.

Table of Contents

Abstract i

Abrégé ii

Acknowledgements iii

List of Tables vii

1 Introduction **1**

1.1 Outline of the Thesis 2

1.2 Statement of Contributions 3

2 Background and Literature Review **4**

2.1 Graphs 4

2.1.1 Static Graphs 4

2.1.2 Dynamic Graphs 5

2.2 Graph Representation Learning 6

2.2.1 Static Graph Representation Learning Methods 8

2.2.2 Dynamic Graph Representation Learning Methods 13

2.3 Graph Embedding Stability 17

2.3.1 Graph Embedding Stability in Static Settings 18

2.3.2 Graph Embedding Stability in Temporal Settings 21

3 Methodology **26**

3.1 Problem Definition 27

3.2 Metric Design 27

3.3 Metric Evaluation . . . 28
3.3.1 Properties . . . 28
3.3.2 Synthetic Scenarios . . . 29
3.4 Index Components . . . 30
3.4.1 Graph–Change Measures . . . 30
3.4.2 Representation Change Measures . . . 33
3.4.3 Alignment Kernels . . . 34
3.5 Representation Alignment . . . 36

4 Experiments **37**
4.1 Experimental Setup . . . 38
4.1.1 Models . . . 38
4.1.2 Synthetic Datasets . . . 39
4.2 Index Validation . . . 40
4.2.1 Results Per Each Scenario . . . 40
4.2.2 Aggregated Results Across Scenarios . . . 41
4.3 Model Comparison Using Validated Index . . . 43
4.3.1 Results Across Scenarios . . . 44
4.3.2 Results in Real Dataset . . . 47

5 Discussions and Conclusion **48**
5.1 Implications . . . 49
5.2 Limitations . . . 50
5.3 Future Work . . . 50
5.4 Conclusion . . . 51

A **52**
A.1 Index validation - Integrity gap and win statistics for all indexes before and after Procrustes Alignment . . . 52

A.2 Integrity Scores after Procrustes Alignment 56

List of Tables

4.1 Synthetic Dataset Statistics 40
4.2 Integrity scores in merge setting. 42
4.3 Integrity scores in move setting. 43
4.4 Integrity scores in periodic setting. 44
4.5 Integrity-gap and win statistics for the chosen index (Euclidean, Euclidean, Pearson) in both not aligned and aligned versions. The first column shows if representation alignment is performed or not. 45
4.6 Integrity scores for each model on the three synthetic scenarios 46
4.7 Integrity scores in CanParl Dataset. 47

A.1 Integrity-gap and win statistics for every index in nonaligned version. . . . 52
A.2 Integrity-gap and win statistics for every index in aligned version. 54
A.3 Integrity scores in merge setting, when representations are aligned between timestamps. 57
A.4 Integrity scores in move setting. 58
A.5 Integrity scores in periodic setting. 59
A.6 Integrity scores for each model after representation alignment. 60

Chapter 1

Introduction

Many real-world networks-such as transportation systems, trade networks, and cryptocurrency networks-evolve over time. When the graph structure changes across time, the network is modeled as a dynamic graph. As deep learning and network science continue to advance, dynamic graph learning has become essential for analyzing such complex, time-varying systems [38]. Representation integrity refers to how truthfully the learned representations reflect both the structural and semantic patterns of the input data, independent of any specific downstream task. This thesis focuses on assessing representation integrity in dynamic graph learning methods [64], emphasizing its role in capturing reliable and meaningful network dynamics.

Representation integrity encompasses both trustworthiness and interpretability, serving as a foundation for understanding the quality and reliability of learned representations [16, 20]. We identify two key aspects of representation integrity in dynamic graph models: Static Integrity, which evaluates how well node embeddings at a given time reflect the graph's structure and semantics, and Temporal Integrity, which assesses how faithfully changes in node embeddings over time reflect genuine structural and semantic changes in the graph, such as shifts in behavioral patterns.

In this work, we propose indexes to quantitatively measure Temporal Integrity in dynamic graph learning models. This framework enables interpretable and diagnostic as-

sessment at the representation level rather than relying solely on specific downstream tasks. It provides a complementary evaluation tool to standard metrics for temporal graph learning models, addressing some evaluation challenges identified in recent studies [7, 57].

Improving the truthfulness of dynamic graph representations has practical implications across a range of domains. In anomaly detection, models with high representation integrity can more reliably identify unusual changes in node behavior or relationships, which is critical in cybersecurity and fraud detection. In trend analysis-such as within social networks or financial markets-temporally consistent embeddings help track evolving patterns with greater accuracy. Additionally, in predictive tasks like link prediction or node classification, high-integrity embeddings yield more interpretable and consistent outputs over time. Enhanced temporal coherence in such embeddings can lead to better detection of behavioral shifts and more accurate forecasting. By focusing on temporal representation integrity we enable more robust and meaningful analysis of dynamic networks.

By focusing on both dynamic and static aspects of representation integrity, researchers can develop more robust and reliable dynamic graph representation learning models. Future research should also expand this evaluation framework across a wider range of models and datasets to further explore the broader implications of representation integrity. This research aims to pave the way for more robust and reliable analyses of complex, evolving systems, ultimately enhancing our ability to understand and predict the behavior of dynamic networks across various domains.

1.1 Outline of the Thesis

Chapter 2 surveys the theoretical background-graph theory, static and dynamic embedding methods, and earlier notions of stability-to show why integrity is a natural next step. Chapter 3 formalizes temporal integrity, proposes a modular metric design space

that combines graph-change and representation-change measures with an alignment kernel, states the properties a sound index must satisfy, and details three synthetic stress-test scenarios (Gradual Merge, Abrupt Move, Periodic Re-wiring). Chapter 4 validates forty-two candidate indexes against these properties and scenarios, selects the best metric, and deploys it to compare eight representative dynamic-embedding models on both synthetic and real data, also examining how integrity correlates with link-prediction performance. Finally, Chapter 5 interprets the empirical results, highlights scenario-specific strengths and weaknesses, and outlines avenues for future work, while an appendix presents complete numerical tables for transparency.

1.2 Statement of Contributions

This work makes four main contributions to temporal graph representation learning:

- We introduces *representation integrity* as a precise, task-agnostic notion of truthfulness for dynamic embeddings, separating static and temporal quality.
- We propose a modular evaluation framework that factors an integrity index into a graph-change measure, a representation-change measure, and an alignment kernel, supported by formal properties.
- We screen forty-two candidate indexes and identify the *Euclidean graph change, Euclidean representation change, Pearson alignment* metric as one satisfying all properties across all stress tests.
- We provide the first comprehensive integrity-based comparison of eight dynamic-embedding models on controlled synthetic scenarios and a real parliamentary-voting dataset, revealing clear scenario-specific performance patterns and a strong positive correlation between integrity and downstream link-prediction accuracy.

Chapter 2

Background and Literature Review

2.1 Graphs

Graphs are a powerful data structure used to represent complex relationships between entities. A graph consists of a set of nodes (also called vertices) and a set of edges connecting pairs of nodes. In a graph, nodes represent entities, while edges represent the connections between these entities. [53]. This structure is particularly useful in various domains such as social networks [25], biological networks [28], transportation systems [48], and citation networks [47]. For instance, in social networks, nodes can represent individuals, and edges can represent friendships or interactions between them.

2.1.1 Static Graphs

A static graph is a type of graph in which the set of nodes and the set of edges connecting these nodes are fixed and do not change over time. Formally, a static graph is represented as $G = (V, E)$ where $V = \{v_1, v_2, ..., v_{|V|}\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges [72].

Stochastic Block Model

The Stochastic Block Model (SBM) is a generative model to create synthetic graphs that often exhibit community structure [33]. In this model, each node belongs to a specific community, and the likelihood of an edge existing between any two nodes is determined by their community memberships. The SBM is characterized by several parameters: the number of nodes n, the number of communities C, a probability vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_C)$ that defines the distribution of nodes across the communities, and a symmetric matrix $W \in \mathbb{R}^{C \times C}$ with entries in the range $[0, 1]$, which specifies the probabilities of connections between nodes in different communities.

A graph G with community labels X is generated under the SBM model SBM(n, α, W) as follows: the community label vector X is an n-dimensional vector where each component is independently drawn according to the distribution α. The graph G is then constructed by connecting any two nodes i and j with a probability W_{X_i, X_j}, independently of other pairs of nodes.

2.1.2 Dynamic Graphs

Traditionally, research in graph theory and graph-based machine learning has concentrated on static graphs, where the structure remains unchanged over time [34]. However, many real-world systems are inherently dynamic, characterized by evolving nodes and edges that change continuously or discretely over time [34]. In the literature, the terms "dynamic" and "temporal" are often used interchangeably to describe evolving graphs or data over time. In this work, we will follow the same convention and use both terms interchangeably.

Dynamic graphs are essential for understanding complex systems in fields such as social networks, communication networks, biological systems, and recommender systems [35]. For instance, in social media, communication events such as emails and text

messages are streaming while friendship relations evolve. In recommender systems, new products, new users and new ratings appear every day.

There are two primary forms of dynamic graphs: discrete-time dynamic graphs (DTDGs) and continuous-time dynamic graphs (CTDGs) [38].

- A discrete-time dynamic graph (DTDG) is a series of graph snapshots taken at regular intervals. Formally, we define a DTDG as a set $\{G_1, G_2, \ldots, G_T\}$, where $G_t = \{V_t, E_t\}$ represents the graph at snapshot t. Here, V_t is the set of nodes, and E_t is the set of edges at time t. The term dynamic graph encompasses both discrete-time dynamic graph (DTDGs) and continuous-time dynamic graphs (CTDGs).

- A continuous-time dynamic graph (CTDG) can be represented as a pair (G, O), where G denotes a static graph that captures the initial state of the dynamic graph at time t_0, and O is a set of observations or events. Each observation in O is a tuple of the event's information and timestamp. The event types can include actions such as edge addition, edge deletion, node addition, node deletion, node splitting, node merging, and others.

2.2 Graph Representation Learning

Before the advent of deep learning, graph representation primarily relied on traditional methods such as adjacency matrices, feature extraction [10, 60], Laplacian eigenmaps [9], and spectral clustering [54]. While these approaches laid the foundation for graph analysis, they often struggled with scalability and comprehensive information capture, especially for large and complex graphs [39]. Specifically, these methods faced challenges in processing graphs with millions of nodes and edges, and often failed to capture higher-order structural information [31].

In order to use graphs in different downstream applications, it is important to represent them effectively. Advancements in data availability and computational resources have led to the development of more flexible graph representation methods, particularly

graph embedding techniques. These newer approaches, including node2vec [26], DeepWalk [56], and Graph Convolutional Networks (GCNs) [42], offer improved scalability and the ability to capture complex structural information.

Graph embedding methods project graph elements (nodes, edges, subgraphs) into a lower-dimensional space while preserving important structural and semantic information [14,31]. Graph embeddings benefit various fields, from computational social science to computational biology [31]. In the literature, the terms "representation" and "embedding" are often used interchangeably when referring to the mapping of nodes or other graph elements to a vector space. In this work, we will follow the same convention and use both terms interchangeably.

Node representation learning is a prominent part of graph representation learning. It refers to the process of learning to map nodes in a graph to dense vector representations in a low-dimensional space. Node representations, or embeddings, capture the structural and semantic information of the nodes, allowing for efficient computation and analysis of graph data.

Formally, given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, node embedding aims to learn a function $f : V \to \mathbb{R}^d$ that maps each node $v \in V$ to a d-dimensional vector, where d is typically much smaller than $|V|$. Node embeddings have several common applications, including graph visualization, clustering, node classification, and link prediction [31].

We will explore relevant works in this area in two subsections: static and dynamic graph representation learning methods. Both categories can be further divided into traditional graph embedding and graph neural network (GNN) based approaches [39]. Traditional methods capture graph information using techniques such as random walks, matrix factorization, and non-GNN deep learning algorithms [39]. In contrast, GNN-based methods leverage neural network architectures specifically designed for graph-structured data.

2.2.1 Static Graph Representation Learning Methods

These methods are divided into traditional graph embedding methods and Graph Neural Network(GNN)-based methods. Traditional methods include dimension-reduction-based methods, random walks, factorization methods, and non-GNN deep learning [39].

- Factorization-based methods capture the relationships between nodes using a matrix representation and then decompose this matrix to derive node embeddings [8] [60]. Commonly used matrices include the node adjacency matrix, Laplacian matrix, node transition probability matrix, and Katz similarity matrix, among others. The specific method for matrix factorization depends on the properties of the matrix. For instance, if the matrix is positive semi-definite, like the Laplacian matrix, eigenvalue decomposition can be applied. For more general, unstructured matrices, gradient descent methods can be used to compute the embeddings efficiently in linear time.

- Random walk-based methods generate node embeddings by utilizing the statistics of random walks. The key innovation of these approaches is to optimize the embeddings such that nodes frequently appearing together in short random walks over the graph have similar embeddings. In contrast to the deterministic similarity measures used in factorization-based methods, random walk-based approaches rely on a more flexible, stochastic notion of node similarity. Prominent examples of this category include DeepWalk [56] and node2vec [26].

- Non-GNN deep learning methods utilize various techniques outside of GNN architectures to learn embeddings. SDNE [70], for instance, is based on an autoencoder which tries to reconstruct the adjacency matrix of a graph and captures nodes' first-order and second-order proximities. Other notable works in this category include [1], [3], and [21].

Graph Neural Networks

Graph Neural Network (GNN) models represent a category of neural networks specially crafted to process graph data [61]. A GNN is a deep learning model which generates a node embedding by aggregating the node's neighbors embeddings. The GNN's intuition is that a node's state is influenced by its interactions with its neighbors in the graph [30, 42]. Since their inception, GNNs have seen rapid development in various architectures, applications, and theoretical studies [78].

Architecture GNNs can generate node representation vectors by stacking several GNN layers. Let h_i^l represent the node embeddings for node i at layer l. Each GNN layer takes as input the nodes embeddings. The node representations for node i at each layer $l+1$ are updated using the following formula:

$$h_i^{l+1} = f(h_i^l, \sum_{j \in N(i)} g(i,j))$$

where f and g are learnable functions and $N(i)$ are the neighbors of node i. h_i^0 are the initial features of node i [73]. At each layer, the embedding of the node i is obtained by aggregating the embeddings of the node's neighbors. After passing through L GNN layers, the final representation of node i is h_i^L, which is the aggregation the node's neighbors of L hops away from the node.

Many GNN models have been developed in recent years. Here, we highlight some key models that have made significant contributions to advancing graph-based learning techniques.

- Graph convolution neural network (GCN): GCNs are a type of neural network specifically designed to operate on graph-structured data [42]. They extend the principles of convolutional neural networks (CNNs), which are widely used for grid-like data such as images, to non-Euclidean domains like graphs. GCNs work by performing convolution operations directly on the graph, aggregating informa-

tion from a node's neighbors to update its representation. This process, known as neighborhood aggregation or message passing, allows GCNs to capture the local structure and features of the graph effectively.

For a static graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, the GCN layer can be formulated as [42]:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{2.1}$$

Where:

- $H^{(l)}$ is the matrix of node features at layer l
- $\tilde{A} = A + I_N$ is the adjacency matrix with added self-connections (I_N is the identity matrix of size $N \times N$)
- $\tilde{D}$ is the degree matrix of $\tilde{A}$
- $W^{(l)}$ is the weight matrix for layer l
- $\sigma(\cdot)$ is a non-linear activation function

For dynamic graphs, where the graph structure and node features evolve over time, the formulation extends to incorporate temporal dependencies. Given a sequence of graphs $G = \{G^1, G^2, ..., G^T\}$, the dynamic GCN layer at time t can be expressed as [51]:

$$H_t^{(l+1)} = \sigma(\tilde{D}_t^{-\frac{1}{2}}A_t\tilde{D}_t^{-\frac{1}{2}}H_t^{(l)}W^{(l)} + H_{t-1}^{(l)}U^{(l)}) \tag{2.2}$$

Where:

- $H_t^{(l)}$ represents the node features at time t and layer l
- A_t and $\tilde{D}_t$ are the adjacency and degree matrices at time t
- $W^{(l)}$ is the spatial weight matrix

- $U^{(l)}$ is a temporal component (could be a weight matrix or some temporal function).
- σ is a non-linear activation function (e.g., ReLU).

This formulation allows the model to capture both spatial dependencies within each graph snapshot and temporal dependencies across different time steps, making it suitable for analyzing evolving graph structures.

- Graph attention network (GAT): The Graph Attention Network is an advanced neural network designed for graph-structured data, which addresses limitations of earlier methods that relied on graph convolutions or their approximations. Unlike traditional approaches, GAT [68] leverages masked self-attentional layers to address the shortcomings of prior methods. By stacking layers in which nodes are able to attend over their neighborhoods' features, it enables (implicitly) specifying different weights to different nodes in a neighborhood. GATs are well-suited for both inductive and transductive tasks due to their ability to dynamically assign attention weights to neighboring nodes, enabling effective learning from local graph structures and generalization to unseen nodes or edges.

- GraphSAGE: GraphSAGE [30] is a framework in the field of Graph Neural Networks (GNNs) that addresses the challenges associated with learning on large-scale graphs.

  GraphSAGE operates by first selecting a set number of neighboring nodes for each node to focus on, rather than using the entire graph. It then aggregates information from these selected neighbors, using methods like averaging their features. This aggregated information is combined with the node's own features to produce a new, updated representation for that node. This process enables GraphSAGE to handle large graphs efficiently and to create useful embeddings even for new nodes not seen during training. GraphSAGE allows for various aggregation functions, includ-

ing mean, LSTM, and pooling operations, to capture different aspects of the node's local structure.

Training Setting GNNs can be trained in supervised, semi-supervised or unsupervised frameworks [79]. In both supervised and semi-supervised frameworks, a variety of prediction tasks, such as node classification and link prediction, can be employed to train the model, targeting nodes, edges, or entire graphs. For example, in node classification within a supervised framework, node embeddings generated by the output of GNN layers are passed through an MLP or softmax layer to produce predictions. A typical loss function for training a GNN in a binary node classification task is the cross-entropy loss, formulated as:

$$L = \sum_{i \in V(train)} y_i \log(\sigma(h_i^T \theta)) + (1 - y_i) \log(1 - \sigma(h_i^T \theta))$$

where h_i is the embedding of node i which is the output of the last layer of GNN, h_i^T is the transpose of h_i, and y_i is the true class label of the node and θ are the classification weights.

Downstream Applications With the abundance of graph-structured data in science and society, GNNs have found wide applicability, with meaningful impact spanning diverse domains, which encompass recommender systems [65], computer vision [45], natural language processing [75], program analysis, software mining, bioinformatics [47], anomaly detection [40], and urban intelligence [49], among others. The fundamental prerequisite for GNN utilization is the transformation or representation of input data into a graph-like structure. In the realm of graph representation learning, GNNs excel in acquiring essential node or graph embeddings that serve as a crucial foundation for subsequent tasks [79]. The downstream applications where GNNs have proven effective can be categorized as follows:

- Node classification involves assigning labels to nodes by leveraging the labels of their neighboring nodes. This task is typically approached using semi-supervised learning, where only a subset of the graph is labeled. The algorithm learns from the labeled nodes and generalizes to label the remaining nodes.

- Graph classification entails categorizing entire graphs into distinct classes, analogous to image classification but applied to graph structures. This method has diverse applications, such as cancer subtype classification in bioinformatics [47] or text classification in natural language processing [75].

- Graph visualization is a field intersecting geometric graph theory and information visualization, focusing on the visual representation of graphs [46]. Effective graph visualization techniques reveal underlying structures and anomalies within the data, aiding users in comprehending complex graph-based information [77].

- Link prediction aims to infer the existence of connections between entities within a graph. This technique is crucial in social networks for deducing potential social interactions or suggesting new connections, such as friends or collaborators [**?**]. Additionally, link prediction is applied in recommender systems [65] and in identifying potential criminal associations.

- Graph Clustering is the process of grouping the nodes of the graph into clusters, taking into account the edge structure of the graph in such a way that there are several edges within each cluster and very few between clusters. Graph Clustering intends to partition the nodes in the graph into disjoint groups [11] [67].

2.2.2 Dynamic Graph Representation Learning Methods

Dynamic graph learning methods are divided into traditional dynamic graph embedding methods and Dynamic Graph Neural Network(DGNN)-based methods. Traditional dynamic graph representation learning approaches can be categorized into four main types:

Aggregation based, Random walk based, Non-GNN based deep learning based, and temporal point process based methods. These approaches aim to capture various dynamic behaviors in networks such as topological evolution and temporal link prediction.

- **Aggregation based methods:** Aggregation-based dynamic graph embedding methods generate embeddings by combining information from dynamic graphs. They generally fall into two categories. The first involves aggregating temporal features, where the evolution of the graph is condensed into a single static graph and static graph embedding methods are applied on the single graph to generate the embeddings. Techniques such as summing adjacency matrices or using weighted sums that emphasize more recent snapshots are applied. However, this approach loses detailed temporal information, such as the specific timing of edge creation. Factorization-based models also fit into this category by representing the sequence of graphs as a three-dimensional tensor and applying factorization techniques to generate embeddings.

  The second category involves aggregating static embeddings. In this approach, static embedding methods are first applied to each graph snapshot individually. The resulting embeddings are then combined into a single matrix, often using a decay factor to prioritize more recent snapshots. Alternatively, time-series models like ARIMA may be used to predict embeddings for future snapshots based on the sequence of previous graphs [15].

- **Random walk based methods:** These methods extend traditional random walk techniques to dynamic graphs. By simulating random walks over time, they capture the evolving connectivity patterns. The embeddings are updated based on the sequences of nodes visited in these walks, which helps in capturing temporal dependencies [6].

- **Non-GNN based deep learning methods:** Non-GNN based deep learning methods for dynamic graph embedding, such as DynGEM [24], Dyn-VGAE [50], and

Dyngraph2vec [23], utilize models like RNNs and autoencoders to capture temporal graph evolution. DynGEM generates embeddings by initializing each snapshot with the previous time step's embeddings, adapting to graph growth. Dyn-VGAE uses a loss function combining VGAE [41] loss and KL divergence to maintain temporal consistency. Dyngraph2vec processes adjacency matrices from past snapshots to generate embeddings, with variants using different network architectures. These methods focus on preserving both structural integrity and temporal patterns as the graph evolves.

- **Temporal point process based:** Temporal point process-based methods are a class of dynamic graph embedding techniques that model the interactions between nodes as stochastic processes occurring over time [6]. These methods assume that the formation and evolution of graph structures are driven by underlying probabilistic events, which can be captured and analyzed using temporal point processes.

  In this framework, the occurrence of edges between nodes is treated as discrete events happening at specific time points, allowing the model to predict when and how these interactions will occur in the future. By leveraging the rich statistical properties of point processes, these methods can effectively capture the temporal dynamics of node interactions, offering insights into the underlying mechanisms driving graph evolution.

Dynamic Graph Neural Networks

Dynamic Graph Neural Networks (DGNNs) are designed to manage graphs that change over time. They process sequences of graph snapshots, each representing the graph at a specific moment. For each snapshot, DGNNs create embeddings that reflect both the current state of the graph and its historical changes.

DGNNs employ dynamic message-passing algorithms that not only consider the current graph structure but also integrate information from previous snapshots. This approach allows nodes to exchange information across different time steps.

To capture temporal changes, DGNNs aggregate information from past snapshots. This can involve techniques such as recurrent neural networks (RNNs) or attention mechanisms, which weigh more recent data more heavily. As the graph evolves, DGNNs continuously update node embeddings to reflect both new and historical information, enabling the model to adapt to new patterns and trends.

In addition to processing current data, DGNNs use learned temporal sequences to predict future changes in the graph, such as emerging edges or evolving node attributes.

DGNNs are applied in various fields, including social networks for tracking user interactions, traffic systems for forecasting congestion, financial markets for predicting transaction trends, and biological networks for studying process changes over time. Overall, DGNNs extend traditional graph neural networks by integrating temporal information, allowing them to handle and predict dynamic changes in graphs.

Dynamic Graph Embedding Applications Dynamic graph embeddings are used to represent graphs that evolve over time, capturing both structural and temporal changes. These embeddings find applications across various domains, providing insights and predictions based on dynamic processes.

- **Social Networks:** In social networks, dynamic graph embeddings track changes in user interactions and relationships. They help in understanding evolving social dynamics, predicting user behavior [17], and identifying emerging communities or influential nodes.

- **Traffic Systems:** For traffic systems, dynamic graph embeddings analyze and forecast traffic flow [37] and congestion patterns. They model the changing road network, predict traffic jams, and optimize routing based on historical and real-time data.

- **Financial Markets:** In financial markets, these embeddings monitor and predict shifts in transactions and market trends [58]. They assist in detecting anomalies, forecasting stock prices, and understanding the flow of financial activities over time.
- **Biological Networks:** In biological networks, dynamic graph embeddings study changes in protein interactions, gene expressions, and other biological processes. They help in understanding disease progression, drug interactions, and the temporal evolution of biological systems [59].
- **Recommendation Systems:** Dynamic graph embeddings enhance recommendation systems by adapting to users' changing preferences and interactions [65]. They improve recommendations by capturing evolving user-item relationships and trends.
- **Infrastructure Management:** For infrastructure management, these embeddings are used to monitor and predict changes in utility networks, such as water or electricity grids. They help in optimizing maintenance schedules and detecting faults and resource allocation [71].

Overall, dynamic graph embeddings provide valuable insights and predictions by capturing the temporal evolution of graph structures and relationships, enabling accurate modeling and understanding of dynamic systems in various domains.

2.3 Graph Embedding Stability

Node embeddings have emerged as a powerful technique for representing nodes in complex networks. They capture the structural and semantic information of nodes in a low-dimensional vector space, enabling various downstream tasks such as link prediction, node classification, and recommendation systems. However, the rapid evolution of these methods has also introduced new challenges, particularly in the realm of temporal stability for dynamic graph representations.

This thesis focuses on addressing the stability issues in dynamic graph representation learning, a critical aspect that has emerged as graph embedding methods have been applied to time-varying networks. By investigating temporal consistency and structural fidelity, we aim to enhance the reliability of evolving network models, contributing to the ongoing advancement of graph representation techniques.

A key trait of a reliable graph representation method is that it should exhibit embedding stability, which can be defined for a static graph through different runs or parameter initializations, etc or for a dynamic graph to analyze embedding stability through time. Understanding the stability of node representation models is crucial for designing effective node embedding methods and maximizing their utility in network analysis.

2.3.1 Graph Embedding Stability in Static Settings

Works in this section examine what we term ”node embedding stability” in a static context, focusing on consistency across different runs or configurations of node embedding models. While these studies use similar terminology to our approach, we actually consider them under Sensitivity and Robustness Analysis as they analyze how sensitive the embeddings are to changes in model parameters or initialization, and how robust they are across different runs. This distinction is important as it differentiates between static and dynamic stability concepts. We will review key works in this area, highlighting their contributions to understanding embedding stability in static graph settings.

In [69], authors define the stability of network embedding as the invariance of the nearest neighbors of nodes in different embedding spaces that have been yielded from different runs. They investigate the potential influence factors on stability, specifically examining network structure properties and algorithm properties. This work sheds light on the fact that instability has remarkable impacts on the performance of network embedding in downstream applications.

In [62], authors define embedding stability informally as the difference between two embeddings due to randomness in the embedding algorithm and introduce three simi-

larity metrics to measure geometric stability and one metric to assess method's stability under randomness with respect to their performance in downstream tasks such as node classification and link prediction. They look into the influence of node centrality, network size and density on the geometric stability of node embeddings. Also in [43], authors assessed the instability of graph neural network predictions with respect to random initialization, model architecture, data, and training setup. Their experiments show that multiple instantiations of popular GNN models trained on the same data with the same model hyperparameters result in almost identical aggregated performance, but display substantial disagreement in the predictions for individual nodes. Authors show that instability of GNNs is reflected in their internal representations and maximizing model performance almost always implicitly minimizes prediction instability.

In [2] and [76], representation stability is measured by the instability score, defined as the percentage of test nodes for which predicted label changes when random noise is added to node attributes. In [2], authors show that representations generated by their method, NIFTY, are stable, where an encoder function is considered stable according to the notion of Lipschitz continuity.

In [4], authors evaluate their proposed model, RECS, in terms of representation learning stability, through two measures. (1) Representation Stability, by verifying the similarity of the learned vectors across different independent runs of the algorithms, and (2) Performance Stability, where they use embeddings from different runs in a classification task and we measure the variation in the classification performance. Ideally, a robust embedding should satisfy both criteria.

In [63], authors consider a graph $G = (V, E)$ with $|V|$ nodes and $|E|$ relationships. Each node v_i in the graph has a final embedding $z_i \in \mathbb{R}^{1\times d}$. They denote the embeddings of all nodes in the graph as $Z \in \mathbb{R}^{|V|\times d}$. They denote the embeddings from configuration k as Z^k, let N denote the overall number of configurations. The authors present the Graph Gram Index (GGI), described in Algorithm 1, as a metric for evaluating the stability of embeddings across various configurations.

Algorithm 1 Graph Gram Index (GGI) [63]

procedure GGI($Z_1, Z_2, \ldots, Z_N$)
  Input: $Z_1, \ldots, Z_N$
  Output: Stability index $s \in [0, 1]$
  for $l \in [1, \ldots, N]$ **do**
    Center and normalize Z_l
    $S_l = A \circ (Z_l Z_l^T)$
    $s_l = \frac{1}{2|E|} \sum_{i,j \in |V| \times |V|} S_l[i, j]$
  end for
  $s = \text{std}(s_l : l \in [1, \ldots, N])$
end procedure

The Gram matrix $Z^l(Z^l)^T$ is used to capture the covariance information between embeddings z_i^l and z_j^l, avoiding the need for alignment. Intuitively, this is achieved because they compare within one set of embeddings to generate a structure summary, and compare the summaries across configurations. By applying a Hadamard product between the adjacency matrix and the Gram matrix, they focus on node pairs that are actual edges in the graph, ensuring that a stable model maintains the similarity of embeddings for these node pairs across different embedding spaces, regardless of any equivariant transformations.

In the context of word embeddings, there are related works which also look into the stability of embeddings. In particular, Borah et al. [12] investigate the stability of word embedding methods and its implications for natural language processing tasks. The study focuses on three word embedding methods: Word2Vec, GloVe, and fastText, evaluating their stability across multiple runs using intrinsic evaluation based on word similarity. The authors define word embedding stability in terms of the consistency of nearest neighbors for a given word across different embedding spaces generated from multiple runs with varying training parameters. Specifically, they measure stability as the number of shared nearest neighbors that a word's embeddings have in two different spaces, averaged over the entire vocabulary, as below:

$$\text{Stability}(A) = \frac{1}{\text{VocabSize}} \sum_{i=1}^{\text{VocabSize}} \text{stability}(w_i) \tag{2.3}$$

$$\text{where stability}(w) = \frac{|KNN_{E_1}(w) \cap KNN_{E_2}(w)|}{k} \tag{2.4}$$

Here, $KNN_E(w)$ represents the set of top k nearest neighbors of word w in the embedding space E. The research examines four diverse datasets and explores the effect of word embedding stability on downstream tasks such as clustering, POS tagging, and fairness evaluation. Their findings indicate that among the three word embedding methods, fastText demonstrates the highest stability, followed by GloVe and Word2Vec. However, they conclude that word embedding methods are not completely stable. The study highlights the importance of considering word embedding stability in NLP applications and provides insights into the factors affecting embedding consistency across different runs.

Similarly, in [5], authors work on identifying the factors that create instability and measure statistical confidence in the cosine similarities between word embeddings trained on small, specific corpora. More specifically, the Jaccard coefficient for the n most similar words is used to measure stability in this work. This measurement for stability was also used in [32] to compare word embedding algorithms and evaluate their stability and accuracy. Also, [18] and [22] are additional studies that evaluate word embeddings with a similar concept of temporal stability as previously discussed works in this area.

2.3.2 Graph Embedding Stability in Temporal Settings

Temporal node embedding methods are designed to capture the dynamic nature of evolving networks by representing nodes in a way that reflects their changing relationships over time. These methods typically process sequences of network snapshots or continuous event streams to model how node representations evolve. By incorporating temporal information, these embeddings offer a more sophisticated understanding of network

dynamics, which can lead to more accurate predictions and analyses in various applications [74].

A critical aspect of dynamic node embedding methods is their temporal stability [16, 20]. Temporal stability refers to the consistency between changes in node embeddings and changes in the structure of the dynamic graph across different time steps. In temporal networks, embedding stability is crucial for ensuring robustness and reliability in downstream tasks [52]. Significant fluctuations in node embeddings between time periods without underlying cause in the dynamics of graph data can impair the performance and generalizability of predictive models and analytical algorithms.

Building on this understanding of temporal stability, our work introduces two key concepts - temporal consistency and structural fidelity - to provide a more comprehensive framework for assessing and improving the reliability of dynamic graph representations.

Temporal consistency measures how well changes in a node's embedding reflect structural changes in the dynamic graph over time. Structural fidelity ensures that the proximity of node embeddings accurately represents their structural relationships at each time step. By examining these aspects, we aim to provide a comprehensive framework for assessing and improving the reliability of dynamic graph representations, addressing crucial challenges in capturing evolving network structures.

In [24], a dynamic graph representation learning method is proposed and authors introduced the concept of stability in dynamic graph embedding in this work for the first time. In this work, first they define relative stability as

$$S(F;t) = \frac{\frac{||F_{t+1}(V_t) - F_t(V_t)||_F}{||F_t(V_t)||_F}}{\frac{||S_{t+1}(V_t) - S_t(V_t)||_F}{||S_t(V_t)||_F}} \tag{2.5}$$

where V_t is the node set at time t, $S_t(\tilde{V})$ is the weighted adjacency matrix of the induced subgraph of node set $\tilde{V} \subseteq V_t$ and $F_t(\tilde{V}) \in \mathbb{R}^{|\tilde{V}| \times d}$ is the embedding of all nodes in $\tilde{V} \subseteq V_t$ of snapshot t.

Then the stability constant is defined as

$$K_S(F) = \max_{\tau,\tau'} |S(F,\tau) - S(F,\tau')| \tag{2.6}$$

In this work, a dynamic embedding F is considered stable as long as it has a small stability constant. The smaller the $K_S(F)$ is, the more stable the embedding F is.

Through experiments they demonstrate the stability of their technique across time and show that their method maintains its competitiveness on evaluation tasks like graph reconstruction, link prediction and visualization.

In [29], authors explore the embedding alignment and its parts, propose metrics to measure alignment and stability, and show their practicality through synthetic experiments. They define the stability error simply as

$$\frac{1}{|V|}\sum_{i=1}^{N} \frac{||n_i^{t+1} - n_i^t||}{||n_i^t|| + ||n_i^{t+1}||} \tag{2.7}$$

where n_i^t is the embedding of node i at time t. This definition gives information about the temporal continuity of the embeddings. In their findings, they do not observe very meaningful differences between static and dynamic methods with respect to the defined stability error. They add that still, dynamic methods usually produce smaller stability errors possibly due to the information sharing between timesteps.

In [27], authors identify a set of classical migration laws and examine them via various methods such as temporal stability analysis. They use the stability error defined in [29], Equation 2.7, to reflect the extent of structural change in dynamic embeddings in this network.

In [20], the authors introduce the concepts of cross-sectional and longitudinal stability and demonstrate that UASE [36] meets both stability criteria. They provide a motivating example to illustrate that these two forms of stability are beneficial for the spatio-temporal representation of nodes. First, cross-sectional stability, which involves attributing the

same position, up to a noise, to nodes that behave similarly at a given time. Second, longitudinal stability, which entails assigning a constant position, up to noise, to an individual node that demonstrates similar behavior across time. They find that existing dynamic latent position models [[55], [13], [80]] tend to trade one type of these stabilities off against the other.

In [16], authors propose that a wide class of static network embedding methods can be used to produce interpretable and powerful dynamic network embeddings when they are applied to the dilated unfolded adjacency matrix. They define two forms of stability: Spatial Stability and Temporal Stability. A dynamic embedding is spatially stable if, for nodes with identical properties at the same time step, their embeddings are exchangeable:

$$P(Y_i^{(t)} = v_1, Y_j^{(t)} = v_2) = P(Y_i^{(t)} = v_2, Y_j^{(t)} = v_1)$$

A dynamic embedding is temporally stable if, for nodes with identical properties at different time steps, their embeddings are exchangeable:

$$P(Y_i^{(u)} = v_1, Y_i^{(t)} = v_2) = P(Y_i^{(u)} = v_2, Y_i^{(t)} = v_1)$$

They provide a theoretical guarantee that these unfolded methods will produce stable embeddings. Furthermore, they considered a range of trivial simulated networks with planted spatial or temporal changes and introduced a hypothesis test which can be used to evaluate how well dynamic embedding methods encode structure. Even in the simplest possible case, unstable dynamic embedding methods are at best conservative, but more often encode incorrect structure. They finally demonstrated the practical advantage that unfolded methods possess over unstable methods by applying them to a pandemic-disrupted dynamic flight network. Here, only the unfolded methods could encode the periodicity and abrupt changes present in the network.

In [52], the authors introduce two notions for ensuring the stability of node embeddings: structural coherence and temporal coherence. Structural coherence is defined as

the condition where, if two nodes exhibit statistically indistinguishable behaviour at a given point in time, their representations at that time are close. That is, if $\Lambda_i(t) = \Lambda_j(t)$, then $\hat{X}_i(t) \approx \hat{X}_j(t)$, where $\Lambda_i(t)$ corresponds to node i's behaviour at time t and $\hat{X}_i(t)$ shows node i's embedding at time t. Temporal coherence is defined as the condition where, if a node exhibits statistically indistinguishable behaviour at two different points in time, its representations at those times are close. That is, if $\Lambda_i(s) = \Lambda_i(t)$, then $\hat{X}_i(s) \approx \hat{X}_i(t)$. In this work authors present a representation learning framework, Intensity Profile Projection, for continuous-time dynamic network data which satisfies both these conditions.

The use of temporal stability in anomaly detection for dynamic graphs offers advantages such as capturing evolving patterns and distinguishing between normal network evolution and unusual events. However, challenges include sensitivity to noise and computational complexity for large-scale graphs. DynGEM [24], for instance, uses Euclidean distance between graph embeddings of consecutive timesteps for event detection, demonstrating one application of this concept.

Chapter 3

Methodology

In this work, we introduce the notion of representation integrity, or truthfulness, in dynamic graph learning, and examine how well dynamic graph learning models align with actual changes in dynamic graphs. Our methodology focuses on an evaluation framework that assesses and compares models based on their temporal integrity. This refers to how accurately the models' learned representations capture the evolving nature of dynamic graphs over time.

First we design a metric to quantify temporal integrity in dynamic graphs which is comprised of three parts: a graph measure, a representation measure, and an alignment kernel. We define a set of integrity metrics in this space and compare and validate them through a set of desired properties, as well as desired behaviour in synthetic scenarios. After designing an integrity metric that satisfies these, we use this selected metric to compare dynamic graph models in terms of their temporal integrity. This evaluation framework built upon our validated indexes can provide a complementary evaluation tool for temporal graph learning models, to help address some of the evaluation challenges mentioned in recent works [7, 57]. Our representation integrity metric can serve as a guiding principle in designing dynamic graph learning models.

3.1 Problem Definition

Let $\mathcal{G} = \{G_t\}_{t=1}^{T}$ be a *discrete-time dynamic graph*, where each snapshot $G_t = (V, E_t)$ consists of a node set V (fixed for simplicity) and a time-varying edge set $E_t \subseteq V \times V$. A dynamic graph embedding model f maps each snapshot G_t to a set of node representations:

$$R_t = f(G_t) \in \mathbb{R}^{|V| \times d},$$

where the d-dimensional row vector $\mathbf{r}_t(v)$ represents node $v \in V$ at time t.

While static graph metrics (e.g., reconstruction loss, neighborhood preservation) are often used to assess embedding quality at individual time steps-what we term "Static Integrity"-this work focuses exclusively on evaluating **Temporal Integrity**: the degree to which changes in the learned representations reflect meaningful changes in the underlying graph over time. A representation is considered to have temporal integrity if it evolves in alignment with graph's dynamics.

3.2 Metric Design

To assess Temporal Integrity in dynamic graph learning models, we propose a general metric design space, consisting of three modular components:

1. A **Graph Change Measure** Δ_t^G, quantifying graph dynamics changes from G_t to G_{t+1}.

2. A **Representation Change Measure** Δ_t^R, capturing how node embeddings change from R_t to R_{t+1}.

3. An **Alignment Kernel** $I(\cdot, \cdot)$, measuring the similarity or consistency between Δ_t^G and Δ_t^R through time.

For each component we consider several candidate measures commonly used in the literature [66]; their definitions are given in Section 3.4.

For each time step t, we compute:

$$\text{TemporalIntegrity}(t) = I(\Delta_t^G, \Delta_t^R),$$

and define **Total Integrity Index (TII)** as the average over time:

$$\text{TII} = \frac{1}{T-1} \sum_{t=1}^{T-1} \text{TemporalIntegrity}(t).$$

3.3 Metric Evaluation

We evaluate a set of indexes composed in the metric design space based on a set of desired properties, and validate them empirically on synthetic data to identify the most effective Integrity Indexes. We then use the validated indexes to compare dynamic graph representation learning models based on how well their embeddings reflect the actual evolution of the graph. Through this formulation, we establish **Temporal Integrity**-as captured by our composable, interpretable metric design space-as a principled criterion for evaluating the integrity of dynamic graph representations.

3.3.1 Properties

For this section, we assume all the graph change and representation change measures can be bounded to $[0, 1]$, that is $\Delta_t^G := \Delta(G_t, G_{t+1}) \in [0, 1]$ and $\Delta_t^R := \Delta(R_t, R_{t+1}) \in [0, 1]$. With this assumption, we propose the following properties that a well-designed integrity index I_t should satisfy:

- **Property 1 (Boundedness).** The integrity score is confined to $[0, 1]$:

$$0 \leq I_t \leq 1 \qquad \forall t.$$

This property guarantees interpretability and comparability across graphs and models.

- **Property 2 (Optimal Alignment)**. The index reaches its maximum when the representation respond perfectly to the true change.

$$\Delta_t^G = \Delta_t^R \Rightarrow I_t = 1$$

- **Property 3 (Upper Anchor).** For a representation that is theoretically stable, I_t must be close to 1:

$$I_t(\text{Stable Model}) \sim 1.$$

If method is not stable, index should be significantly lower.

3.3.2 Synthetic Scenarios

We will empirically test these properties using synthetic dynamic graphs generated with Stochastic Block Models (SBMs), under the following scenarios:

- **Gradual Merge**: Two equally sized communities start almost disconnected; the probability of cross-community edges rises linearly from a low to high probability over T snapshots.

  Motivation: tests a model's ability to track *smooth, long-horizon drift*; representations must evolve continuously without over-reacting to small incremental edits.

- **Abrupt Move**: Three equally sized communities start with high within-community edge probability and only weak ties to other communities (low cross-community edge probability). Throughout the first half of the sequence of length T, the connection probabilities remain as described; at the midpoint, the edge connection probabilities change, such that the nodes in community 1 migrate to community 0 and

adopt the edge connection probabilities of community 0. This new behaviour remains fixed for the rest of the experiment.

Motivation: probes *reactivity to sudden, localized change*; the model must detect and accommodate a sharp structural jump rather than smoothing it away.

- **Periodic Transitions**: Two equally sized communities start with high probability of within-community edges and low cross-community edges. Their interaction pattern follows a smooth cyclic rhythm. Within-community edges become more (or less) likely according to a sinusoid, rising from a low to high probability and back again every $\frac{T}{5}$ snapshots; between-community probabilities vary inversely. The graph therefore alternates between phases of clear modular structure and phases of near mixing, in a regular cycle.

  Motivation: evaluates whether the model retains *temporal memory* and can represent patterns that repeat over long intervals, a capability needed for forecasting seasonality or scheduling effects in real networks.

3.4 Index Components

3.4.1 Graph–Change Measures

To quantify how much the graph evolves from G_t to G_{t+1} we employ three distance measures. Each one is normalized to the unit interval $[0, 1]$ so that integrity scores remain comparable across datasets and scenarios. We include three distances that offer diverse perspectives: adjacency-based (Euclidean), affinity-based (DeltaCon), and spectral.

Scaled Euclidean Adjacency Distance

Let A_t and A_{t+1} be the adjacency matrices on the common node set of size n. We use the Frobenius norm,

$$d_{\mathrm{EUC}}(G_t, G_{t+1}) = \frac{\|A_t - A_{t+1}\|_F}{\sqrt{n(n-1)}},$$

where the denominator is the maximum possible difference (an empty versus a complete graph), yielding a value in $[0, 1]$. The measure captures aggregate edge-level changes and is quick to compute.

DeltaCon Distance

DeltaCon similarity [19] quantifies the resemblance between two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ sharing the same node set.

Let $A_i \in \{0, 1\}^{n \times n}$ be the (possibly weighted) adjacency matrix of G_i and $D_i = \mathrm{diag}(A_i \mathbf{1})$ its degree matrix. For a small diffusion parameter $\epsilon > 0$ we form the node-affinity *Fast Belief Propagation* matrix

$$S_i \;=\; \big(I + \epsilon^2 D_i - \epsilon A_i\big)^{-1}, \qquad i \in \{1, 2\},$$

whose entry $S_i(u, v)$ reflects the ease with which information can flow between nodes u and v in G_i. It can be shown that sij depends on all the r-paths connecting (i, j) [19].

DeltaCon similarity is then defined as

$$\mathrm{DeltaCon}_{\mathrm{sim}}\big(G_1, G_2\big) \;=\; \frac{1}{1 + \sqrt{\sum_{u,v \in V} \big(\sqrt{S_1(u, v)} - \sqrt{S_2(u, v)}\big)^2}}$$

which attains 1 when the graphs are identical and decreases smoothly toward 0 as they diverge.

We convert this similarity into a **graph-change distance** by taking

$$\mathrm{d}_{\mathrm{DeltaCon}}\big(G_t, G_{t+1}\big) \;=\; 1 - \mathrm{DeltaCon}_{\mathrm{sim}}\big(G_t, G_{t+1}\big) \;\in\; [0, 1],$$

so that larger values directly indicate greater change between snapshots.

It can be shown [19, 44] that DeltaCon satisfies several desirable properties: it penalizes changes that disconnect a graph more heavily; in weighted graphs, removing

higher-weight edges increases the distance more; alterations in sparse graphs have greater impact than identical changes in denser graphs; and random modifications yield smaller distance increases than targeted ones.

Laplacian Spectral Distance

Let L_t and L_{t+1} be the Laplacians of A_t and A_{t+1} and $\lambda_1^{(t)} \leq \cdots \leq \lambda_n^{(t)}$ their eigenvalues. We define

$$d_{\mathrm{SPEC}}(G_t, G_{t+1}) = \frac{\left\|\boldsymbol{\lambda}^{(t)} - \boldsymbol{\lambda}^{(t+1)}\right\|_2}{n\sqrt{n-1}},$$

where $\boldsymbol{\lambda}^{(t)}$ is the sorted eigenvalue vector. Norm-2 difference of $\boldsymbol{\lambda}^{(t)}$ and $\boldsymbol{\lambda}^{(t+1)}$, is divided by the maximum possible value $n\sqrt{n-1}$, which is between an empty and a complete graph, to normalize the measure in to $[0, 1]$ internal. This measure captures differences in global connectivity patterns and community structure.

Spectral Distance (first–k eigenvalues)

Let $\widetilde{L}_t$ and $\widetilde{L}_{t+1}$ denote the normalized Laplacians of G_t and G_{t+1}, whose spectra lie in $[0, 2]$. Denote by $0 = \lambda_0^{(t)} < \lambda_1^{(t)} \leq \cdots \leq \lambda_k^{(t)}$ and $0 = \lambda_0^{(t+1)} < \lambda_1^{(t+1)} \leq \cdots \leq \lambda_k^{(t+1)}$ the first k *non-zero* eigenvalues of $\widetilde{L}_t$ and $\widetilde{L}_{t+1}$.

We measure spectral change as the Euclidean distance between these two eigenvalue vectors, normalised by the largest possible distance, $2\sqrt{k}$:

$$d_{\mathrm{SPEC}}(G_t, G_{t+1}) \;=\; \frac{\left\|(\lambda_1^{(t)}, \ldots, \lambda_k^{(t)}) - (\lambda_1^{(t+1)}, \ldots, \lambda_k^{(t+1)})\right\|_2}{\sqrt{k}\;\cdot 2} \;\in [0, 1].$$

Using only the lowest k non-zero modes focuses the distance on *global* structural features-such as coarse community structure and overall connectivity.

3.4.2 Representation Change Measures

To quantify how much a model's node embeddings have changed from snapshot t to $t+1$, we employ two scale-free representation change measures.

In our experiments, we also implement a variant where we perform Procrustes alignment before calculating the representation change. This alignment eliminates discrepancies due to rotations and reflections in the representation space. We describe this variant in the Appendix A.

Row–Cosine Change. Every node representation vector is first ℓ_2-normalised

$$\hat{e}_{t,i} = \frac{(E_t Q^\star)_i}{\|(E_t Q^\star)_i\|}, \qquad \hat{e}_{t+1,i} = \frac{(E_{t+1})_i}{\|(E_{t+1})_i\|},$$

ensuring scale invariance; therefore it captures only directional (i.e., relational) changes, not arbitrary differences in vector magnitude.

For each node we compute the directional drift

$$\Delta r_i^{cos} \;=\; 1 - \cos\big(\hat{e}_{t,i},\, \hat{e}_{t+1,i}\big) \;\in\; [0,1],$$

where 0 means the embedding direction is unchanged and 1 means it has flipped to the opposite pole. The snapshot-level representation change is defined as the bounded average:

$$\Delta R^{cos} \;=\; \frac{1}{n}\sum_{i=1}^{n} \Delta r_i^{cos} \;\in\; [0,1].$$

Row-Unit Euclidean Change. After projecting each node's representation vector onto the unit hypersphere, we measure the half-Euclidean distance:

$$\Delta r_i^{euc} \;=\; \tfrac{1}{2}\left\|\hat{e}_{t,i} - \hat{e}_{t+1,i}\right\| \;\in [0,1].$$

The snapshot-level representation change is defined as the bounded average:

$$\Delta R^{euc} \;=\; \frac{1}{n}\sum_{i=1}^{n} \Delta r_i^{euc} \;\in\; [0,1].$$

3.4.3 Alignment Kernels

We use seven candidate alignment kernels to calculate the integrity index I_t, which measures the agreement between graph changes and representation changes. Each of the alignment kernels considered is bounded within the range $[0,1]$ by design and reaches the theoretical extrema, as stipulated by Property 2 ($\Delta_t^G = \Delta_t^R \Rightarrow I_t = 1$).

Symmetric Deviation Penalty

The kernel

$$I_t^{Sym} = 1 - |\Delta_t^R - \Delta_t^G|$$

computes integrity as the complement of the absolute difference between representation change Δ^R and graph change Δ^G. This linear penalty ensures simplicity and symmetry, treating overreaction and underreaction equivalently.

Gaussian Similarity

Defined as

$$I_t^{Gauss} = \exp\left(-\frac{(\Delta_t^R - \Delta_t^G)^2}{2\sigma^2}\right)$$

this kernel uses a Gaussian function to smoothly penalize misalignment. The sensitivity parameter σ controls the rate of decay in integrity scores as Δ^G diverges from Δ^R.

KL-Divergence Inspired

This kernel is based on the Kullback–Leibler (KL) divergence, a fundamental concept from information theory that quantifies how one probability distribution diverges from a

reference distribution. The standard KL divergence for discrete distributions P and Q is defined as

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x} P(x) \log \left(\frac{P(x)}{Q(x)} \right),$$

and is commonly used to measure the information loss when Q is used to approximate P.

In our context, we adapt this idea to compare the scalar change measures Δ_t^G and Δ_t^R, which are interpreted as probabilities of change in a Bernoulli setting. The kernel is defined as

$$I_t^{KL} = 1 - \left[\Delta_t \log \left(\frac{\Delta_t}{\delta_t + \epsilon} \right) + (1 - \Delta_t) \log \left(\frac{1 - \Delta_t}{1 - \delta_t + \epsilon} \right) \right],$$

where $\epsilon > 0$ is a small constant for numerical stability.

The formulation applies the KL divergence between two Bernoulli distributions, with Δ_t^G as the reference and Δ_t^R as the approximation. The result is then subtracted from 1 to yield a similarity score, so that higher values of I_t indicate better alignment between the representation and the graph change. The smoothing term ϵ ensures the kernel remains well-defined even when Δ_t^G is close to 0 or 1.

Pearson Correlation

A global linear association is captured by the Pearson correlation coefficient $\rho_{\text{pearson}} \in [-1, 1]$:

$$I_t^{\text{pearson}}(\Delta_t^G, \Delta_t^R) \; = \tfrac{1}{2}\big(\rho_{\text{pearson}}(\Delta_t^G, \Delta_t^R) + 1\big) \tag{3.1}$$

Spearman Correlation Spearman's correlation coefficient is a statistical measure of the strength of a monotonic relationship between paired data. Spearman correlation $\rho_{\text{spearman}} \in [-1, 1]$ works by calculating Pearson's correlation on the ranked values of the data.

$$I_t^{\text{spearman}}(\Delta_t^G, \Delta_t^R) \; = \; \tfrac{1}{2}\big(\rho_{\text{spearman}}(\Delta_t^G, \Delta_t^R) + 1\big) \tag{3.2}$$

Time Lagged Cross Correlation Representation models may respond to a structural shock after a short delay. Let $L \geq 0$ be the maximum lag we are willing to tolerate. We scan all shifts within $\pm L$ and keep the best Pearson correlation:

$$I_t^{\text{xcorr}}(\Delta_t^G, \Delta_t^R) \;=\; \tfrac{1}{2}\Big(\max_{|\ell|\leq L} \rho_{\text{pearson}}\big(\Delta^G_{0:T-\ell}, \Delta^R_{\ell:T}\big) + 1\Big). \tag{3.3}$$

In our experiments we set $L = 3$, trading off responsiveness against the risk of hiding genuinely mistimed embeddings.

Dynamic Time Warping Finally, Dynamic Time Warping (DTW) aligns the two trajectories under arbitrary monotone, contiguous warps. Let $\mathrm{DTW}(\cdot,\cdot)$ denote the classic DTW distance and T the sequence length:

$$I_t^{\text{DTW}}(\Delta_t^G, \Delta_t^R) \;=\; \max\Big(0,\, 1 - \frac{\mathrm{DTW}(\Delta_t^G, \Delta_t^R)}{T}\Big). \tag{3.4}$$

Normalising by T ensures scale invariance; truncating negative values at zero interprets extremely large misalignments as complete loss of integrity.

3.5 Representation Alignment

In this section, we describe another approach for computing integrity scores that first applies Procrustes alignment to the node embeddings before integrity score computation. Specifically, to compare embeddings from snapshot t to $t{+}1$, we first align the two embedding matrices $E_t, E_{t+1} \in \mathbb{R}^{n\times d}$ using the optimal orthogonal matrix,

$$Q^\star \;=\; \arg\min_{Q^\mathsf{T}Q=I} \|E_t\, Q - E_{t+1}\|_F,$$

obtained through singular-value decomposition of $E_t^\mathsf{T} E_{t+1}$. This *Procrustes* alignment eliminates arbitrary rotation or reflection that occurs when models are retrained independently at each time step-variations that should *not* count as semantic drift.

Chapter 4

Experiments

The primary aim of our experimental study is twofold: i) To evaluate and select the most reliable integrity indexes for dynamic graph representation learning, using both theoretical and empirical criteria. ii) To compare dynamic graph embedding models in terms of temporal integrity, as measured by the validated indices.

First, we validate the properties of each candidate index by examining its empirical behavior in each of the synthetic scenarios. To guide our analysis, we leverage UASE [36] and IPP [52] models as baselines for further analysis of the candidate indexes, since these models are mathematically proven to maintain integrity. [20]. Based on this analysis, we select a validated index that meets all of our criteria. This allows for a straightforward comparison of temporal integrity of different learning-based models, which we report next. Additionally, we compute the correlation of the integrity scores of these models with their performance in link prediction, which is the common metrics used to evaluate dynamic graph models, to show how a more stable model also generally performs better on the downstream link prediction task.

4.1 Experimental Setup

4.1.1 Models

We evaluate several dynamic graph embedding models, including both learning-based and non-learning-based approaches. The primary models considered are:

- UASE (Unfolded Adjacency Spectral Embedding) [36]: This method extends the random dot product graph model to handle dynamic multilayer networks by jointly embedding adjacency matrices across multiple time steps. It computes a singular value decomposition of the unfolded adjacency matrix to obtain node embeddings that are consistent over time and across similar nodes, making it particularly suitable for spatio-temporal analysis of evolving graphs [20, 36].

- IPP (Intensity Profile Projection) [52]: IPP is a spectral representation learning framework for continuous-time dynamic network data. It works by first smoothing the observed event counts to get a pair-wise intensity function for every edge, then finding the best rank-d subspace that minimizes an intensity-reconstruction error, and finally projecting each node's time-varying intensity profile onto that subspace to obtain its embedding.

- GAT (Graph Attention Network) [68]: This method utilizes the Graph Attention Network (GAT), as described in Section 2.2.1.

- AE (Autoencoder) [24]: This method uses deep autoencoder to learn the representation of each node in the graph.

- dynAE (Dynamic Autoencoder) [23]: This method models the interconnection of nodes within and acrosstime using multiple fully connected layers. It extends Static AE for dynamic graphs.

- dynRNN (Dynamic Recurrent Neural Network) [23] : This method uses sparsely connected Long Short Term Memory(LSTM) networks to learn the embedding.
- dynAERNN (Dynamic Autoencoder Recurrent Neural Network) [23]: This method uses a fully connected encoder to initially ac-quire low dimensional hidden representation and feeds this representation into LSTMs to capture network dynamics.

Each method generates node embeddings of dimension d = 64 for each time step. [1]

4.1.2 Synthetic Datasets

To systematically evaluate integrity indexes, we generate synthetic dynamic graphs using Stochastic Block Models (SBMs) under three controlled scenarios described in 3.3.2:

- **Gradual Merge.** Two communities begin nearly separate; the between-community link probability increases linearly from 0.1 to 0.9 over the $T = 100$ snapshots, so the blocks gradually merge into a single mixed community.
- **Abrupt Move.** Three equal communities evolve steadily with 0.9 within-community edge probability and 0.05 cross-community edge probabilities until the midpoint, when one community suddenly migrates to another community and adopts its connectivity pattern, triggering a sharp structural change.
- **Periodic Transitions.** At each snapshot the within-community connection probability rises and falls in a smooth sine-wave cycle, while the between-community probability changes inversely. With a 20-step cycle repeated over the $T = 100$ timesteps, the graph returns to the same community pattern every 20 steps, mimicking a regular seasonal change.

Each synthetic graph consists of $n = 300$ nodes and approximately $30,000$ edges per timestep (exact number varies by scenario and SBM parameters). Edge probabilities and community sizes are chosen to ensure non-trivial but interpretable dynamics.

[1] We use the implementation of some dynamic graph models accessible in DynamicGEM, A Library for Dynamic Graph Embedding Methods.

Scenario	# Nodes	# Total Edges	T
Merge	300	3,136,569	100
Move	300	1,701,030	100
Periodic	300	2,242,381	100

Table 4.1: Synthetic Dataset Statistics

4.2 Index Validation

We start with 42 candidate integrity configurations and progressively narrow them down to a single index that fulfills all of the properties and expectations.

Combining the property analysis and the UASE or IPP-based empirical evaluation, we identify the subset of indexes that (a) satisfy all properties (including empirical support for property 3), and (b) most frequently recognize UASE or IPP as the best model across scenarios. These indexes are recommended for practical use in dynamic graph representation learning.

We compute the integrity score $I_t = I(\Delta_t^G, \Delta_t^R)$ for each consecutive timestep where $t = 1, \ldots, T-1$ in all synthetic datasets (merge, move, periodic) and summarize it by the time–average.

$$\bar{I} = \frac{1}{T-1} \sum_{t=1}^{T-1} I_t. \tag{4.1}$$

We report the time-averaged integrity scores in tables 4.2 for merge scenario, 4.3 for move scenario, and 4.4 for periodic scenario. For every considered configuration of the integrity index $(\Delta G, \Delta R, \text{kernel})$ in each row, the best, second-best and third-best model scores are highlighted in gold, silver and bronze, respectively.

4.2.1 Results Per Each Scenario

Each candidate integrity index is first assessed against property 3, defined in chapter 3, per each scenario.

An integrity index validates property 3 by assigning a high score to stable methods and demonstrating a sharp contrast between integrity scores of a stable model compared to an unstable baseline.

We use UASE [36] and IPP [52] graph methods as stable models in this experiment, as they are the only models to the best of our knowledge that are mathematically proven to be consistent both cross-sectionally (across nodes) and longitudinally (across time) [20, 52]. Hence their integrity scores should be close to 1.

For the unstable model, a static auto-encoder is trained on the synthetic datasets, then the $T = 100$ snapshot blocks of its embedding are permuted at random. The per-snapshot quality is unchanged, but temporal alignment is destroyed, so its integrity score should be significantly lower than UASE or IPP's.

4.2.2 Aggregated Results Across Scenarios

UASE [36] and IPP [52] are mathematically proven to be temporally coherent [20,52]. We use these models as reference to validate our integrity index: We record in how many of the three synthetic datasets (*merge*, *move*, *periodic*) the index ranks either IPP or UASE as the top model $\left(\text{Win}_{\text{IPP—UASE}} \in \{0, 1, 2, 3\}\right)$. An index that consistently identifies IPP or UASE as the model with the highest integrity is considered more reliable in practice.
Table A.1 evaluates every integrity configuration on two axes:

- **Comparing integrity score gap between stable (UASE and IPP) and unstable models**

  $$\Delta\bar{I}_{\text{UASE}} = \bar{I}_{\text{UASE}} - \bar{I}_{\text{shuffled}}, \qquad \Delta\bar{I}_{\text{IPP}} = \bar{I}_{\text{IPP}} - \bar{I}_{\text{shuffled}}.$$

  A large positive value means the metric assigns a higher score to the theoretically stable representation (UASE or IPP) compared to the unstable baseline.

- **$\text{Win}_{\text{UASE|IPP}}$** -the number of data scenarios (merge, move, periodic) in which the same configuration also ranks either UASE or IPP as the best model (range 0–3).

Table 4.2: Integrity scores in merge setting.

			Non-learning-based		Static			Dynamic			
ΔG	ΔR	kernel	IPP	UASE	GAT	AE	AE-Shuffled	DynAE	DynRNN	DynAERNN	STConv
DeltaCon	Cosine	Sym	0.288	0.814	0.648	0.860	0.870	0.885	0.783	0.764	0.234
DeltaCon	Cosine	Gauss	0.0	0.496	0.278	0.633	0.668	0.718	0.429	0.385	0.0
DeltaCon	Cosine	KL	0.0	0.912	0.488	0.939	0.948	0.955	0.449	0.416	0.002
DeltaCon	Cosine	Pearson	0.795	0.630	0.575	0.345	0.392	0.543	0.533	0.406	0.601
DeltaCon	Cosine	Spearman	0.739	0.580	0.565	0.327	0.38	0.542	0.518	0.417	0.748
DeltaCon	Cosine	Xcorr	0.806	0.671	0.619	0.381	0.463	0.543	0.558	0.538	0.756
DeltaCon	Cosine	DTW	0.288	0.826	0.67	0.879	0.905	0.922	0.817	0.802	0.234
DeltaCon	Euclidean	Sym	0.4	0.767	0.817	0.797	0.799	0.804	0.918	0.918	0.241
DeltaCon	Euclidean	Gauss	0.001	0.324	0.539	0.423	0.429	0.444	0.829	0.836	0.0
DeltaCon	Euclidean	KL	0.136	0.876	0.884	0.898	0.901	0.907	0.976	0.977	0.0
DeltaCon	Euclidean	Pearson	0.799	0.631	0.569	0.343	0.392	0.54	0.533	0.411	0.620
DeltaCon	Euclidean	Spearman	0.738	0.580	0.566	0.325	0.379	0.539	0.518	0.417	0.795
DeltaCon	Euclidean	Xcorr	0.809	0.673	0.614	0.384	0.468	0.54	0.558	0.541	0.807
DeltaCon	Euclidean	DTW	0.4	0.767	0.847	0.8	0.807	0.813	0.940	0.939	0.241
Spectral	Cosine	Sym	0.941	0.415	0.172	0.35	0.344	0.333	0.019	-0.004	0.995
Spectral	Cosine	Gauss	0.921	0.002	0.039	0.0	0.0	0.0	0.0	0.0	0.992
Spectral	Cosine	KL	0.944	0.128	0.177	0.059	0.043	0.046	0.0	0.0	0.996
Spectral	Cosine	Pearson	0.656	0.495	0.474	0.388	0.382	0.583	0.528	0.403	0.572
Spectral	Cosine	Spearman	0.653	0.532	0.507	0.401	0.377	0.580	0.522	0.398	0.697
Spectral	Cosine	Xcorr	0.667	0.521	0.542	0.441	0.464	0.583	0.585	0.507	0.694
Spectral	Cosine	DTW	0.941	0.415	0.172	0.35	0.344	0.333	0.019	0.0	0.996
Spectral	Euclidean	Sym	0.830	0.462	0.38	0.433	0.43	0.425	0.302	0.293	0.989
Spectral	Euclidean	Gauss	0.525	0.002	0.012	0.001	0.001	0.001	0.0	0.0	0.992
Spectral	Euclidean	KL	0.819	0.233	0.173	0.168	0.162	0.149	0.006	0.002	0.991
Spectral	Euclidean	Pearson	0.661	0.496	0.473	0.386	0.382	0.583	0.529	0.407	0.582
Spectral	Euclidean	Spearman	0.653	0.533	0.508	0.4	0.377	0.579	0.522	0.398	0.710
Spectral	Euclidean	Xcorr	0.670	0.523	0.537	0.443	0.469	0.583	0.586	0.51	0.713
Spectral	Euclidean	DTW	0.830	0.462	0.38	0.433	0.43	0.425	0.302	0.293	0.989
Euclidean	Cosine	Sym	0.527	0.947	0.603	0.879	0.874	0.858	0.551	0.528	0.472
Euclidean	Cosine	Gauss	0.009	0.931	0.328	0.706	0.688	0.635	0.061	0.035	0.008
Euclidean	Cosine	KL	0.153	0.993	0.532	0.948	0.950	0.939	0.244	0.173	0.009
Euclidean	Cosine	Pearson	0.871	0.996	0.596	0.507	0.528	0.527	0.546	0.518	0.394
Euclidean	Cosine	Spearman	0.857	0.993	0.608	0.52	0.522	0.57	0.532	0.523	0.601
Euclidean	Cosine	Xcorr	0.922	0.998	0.606	0.537	0.53	0.547	0.546	0.518	0.586
Euclidean	Cosine	DTW	0.527	0.969	0.604	0.894	0.896	0.868	0.551	0.528	0.472
Euclidean	Euclidean	Sym	0.639	0.986	0.843	0.950	0.949	0.945	0.833	0.825	0.479
Euclidean	Euclidean	Gauss	0.062	0.994	0.587	0.914	0.915	0.904	0.544	0.52	0.007
Euclidean	Euclidean	KL	0.66	0.999	0.902	0.991	0.991	0.99	0.93	0.923	0.009
Euclidean	Euclidean	Pearson	0.872	0.996	0.585	0.509	0.53	0.527	0.541	0.52	0.409
Euclidean	Euclidean	Spearman	0.857	0.992	0.609	0.519	0.521	0.571	0.532	0.523	0.656
Euclidean	Euclidean	Xcorr	0.923	0.997	0.594	0.54	0.531	0.546	0.541	0.52	0.676
Euclidean	Euclidean	DTW	0.639	0.995	0.848	0.959	0.962	0.953	0.833	0.825	0.479

We consider a configuration to satisfy the revised properties when

$$\Delta\bar{I}_{\text{UASE}} \geq 0.25, \quad \Delta\bar{I}_{\text{IPP}} \geq 0.25, \quad \text{Win}_{\text{UASE|IPP}} = 3.$$

Under these criteria, the combinations of **Euclidean ΔG + Euclidean or Cosine ΔR + Pearson, Spearman, or XCorr Alignment kernel** pass all three thresholds and are therefore appropriate choices of integrity index. As their performance are on par, we choose **Euclidean ΔG + Euclidean ΔR + Pearson** as the final integrity index for the remain-

Table 4.3: Integrity scores in move setting.

			Non-learning-based		Static			Dynamic			
ΔG	ΔR	kernel	IPP	UASE	GAT	AE	AE-Shuffled	DynAE	DynRNN	DynAERNN	STConv
DeltaCon	Cosine	Sym	0.256	0.686	0.646	0.819	0.814	0.795	0.809	0.806	0.217
DeltaCon	Cosine	Gauss	0.0	0.12	0.303	0.493	0.473	0.412	0.491	0.473	0.0
DeltaCon	Cosine	KL	0.002	0.788	0.452	0.919	0.915	0.898	0.468	0.47	0.006
DeltaCon	Cosine	Pearson	0.832	0.724	0.623	0.51	0.519	0.557	0.534	0.549	0.543
DeltaCon	Cosine	Spearman	0.795	0.225	0.646	0.477	0.511	0.554	0.515	0.545	0.602
DeltaCon	Cosine	Xcorr	0.832	0.724	0.660	0.557	0.609	0.597	0.553	0.549	0.645
DeltaCon	Cosine	DTW	0.256	0.686	0.67	0.819	0.814	0.795	0.864	0.876	0.217
DeltaCon	Euclidean	Sym	0.36	0.692	0.865	0.757	0.755	0.746	0.901	0.905	0.227
DeltaCon	Euclidean	Gauss	0.0	0.127	0.661	0.28	0.272	0.249	0.782	0.802	0.0
DeltaCon	Euclidean	KL	0.02	0.795	0.934	0.866	0.864	0.855	0.969	0.972	0.003
DeltaCon	Euclidean	Pearson	0.846	0.704	0.624	0.513	0.519	0.554	0.536	0.55	0.579
DeltaCon	Euclidean	Spearman	0.797	0.223	0.646	0.478	0.508	0.548	0.515	0.547	0.756
DeltaCon	Euclidean	Xcorr	0.846	0.704	0.659	0.556	0.606	0.596	0.554	0.55	0.666
DeltaCon	Euclidean	DTW	0.36	0.692	0.875	0.757	0.755	0.746	0.903	0.908	0.227
Spectral	Cosine	Sym	0.950	0.520	-0.004	0.386	0.392	0.411	0.024	0.014	0.986
Spectral	Cosine	Gauss	0.942	0.006	0.004	0.0	0.0	0.001	0.0	0.0	0.987
Spectral	Cosine	KL	0.954	0.352	0.067	0.074	0.084	0.128	0.001	0.0	0.988
Spectral	Cosine	Pearson	0.999	0.967	0.521	0.542	0.511	0.53	0.531	0.544	0.511
Spectral	Cosine	Spearman	0.636	0.352	0.547	0.423	0.506	0.473	0.434	0.418	0.559
Spectral	Cosine	Xcorr	0.999	0.967	0.584	0.575	0.649	0.587	0.598	0.544	0.579
Spectral	Cosine	DTW	0.950	0.520	0.0	0.386	0.392	0.411	0.024	0.014	0.986
Spectral	Euclidean	Sym	0.846	0.514	0.307	0.449	0.451	0.46	0.304	0.301	0.977
Spectral	Euclidean	Gauss	0.590	0.005	0.002	0.001	0.001	0.002	0.0	0.0	0.982
Spectral	Euclidean	KL	0.839	0.343	0.083	0.206	0.211	0.23	0.008	0.003	0.980
Spectral	Euclidean	Pearson	0.997	0.955	0.525	0.543	0.513	0.531	0.532	0.544	0.547
Spectral	Euclidean	Spearman	0.640	0.353	0.548	0.422	0.508	0.473	0.434	0.42	0.641
Spectral	Euclidean	Xcorr	0.997	0.955	0.576	0.574	0.644	0.586	0.593	0.544	0.547
Spectral	Euclidean	DTW	0.846	0.514	0.307	0.449	0.451	0.46	0.304	0.301	0.977
Euclidean	Cosine	Sym	0.693	0.877	0.351	0.744	0.749	0.768	0.381	0.371	0.654
Euclidean	Cosine	Gauss	0.125	0.713	0.137	0.248	0.262	0.326	0.007	0.001	0.078
Euclidean	Cosine	KL	0.545	0.969	0.318	0.861	0.867	0.884	0.125	0.065	0.041
Euclidean	Cosine	Pearson	0.976	0.994	0.498	0.552	0.516	0.524	0.536	0.536	0.489
Euclidean	Cosine	Spearman	0.253	0.966	0.444	0.564	0.550	0.53	0.518	0.452	0.391
Euclidean	Cosine	Xcorr	0.976	0.994	0.552	0.582	0.649	0.576	0.598	0.536	0.531
Euclidean	Cosine	DTW	0.693	0.877	0.471	0.771	0.782	0.811	0.386	0.371	0.655
Euclidean	Euclidean	Sym	0.797	0.871	0.665	0.805	0.807	0.816	0.661	0.658	0.664
Euclidean	Euclidean	Gauss	0.402	0.690	0.207	0.432	0.441	0.475	0.098	0.086	0.089
Euclidean	Euclidean	KL	0.877	0.966	0.675	0.922	0.924	0.930	0.748	0.746	0.263
Euclidean	Euclidean	Pearson	0.966	0.990	0.503	0.552	0.517	0.524	0.535	0.536	0.516
Euclidean	Euclidean	Spearman	0.252	0.967	0.445	0.566	0.549	0.533	0.518	0.452	0.183
Euclidean	Euclidean	Xcorr	0.966	0.990	0.546	0.58	0.644	0.574	0.592	0.536	0.516
Euclidean	Euclidean	DTW	0.797	0.871	0.77	0.806	0.809	0.818	0.685	0.658	0.664

der of this thesis. Table 4.5 presents the results for this index in both versions, where representations are either aligned before the integrity score computation or used without modification.

4.3 Model Comparison Using Validated Index

Using the validated integrity metric with composition *Euclidean* ΔG + *Euclidean* ΔR + *Pearson*, we score every model on the three synthetic streams. Table 4.6 reports the time-

Table 4.4: Integrity scores in periodic setting.

			Non-learning-based		Static			Dynamic			
ΔG	ΔR	kernel	IPP	UASE	GAT	AE	AE-Shuffled	DynAE	DynRNN	DynAERNN	STConv
DeltaCon	Cosine	Sym	0.316	0.834	0.642	0.802	0.792	0.726	0.805	0.807	0.205
DeltaCon	Cosine	Gauss	0.0	0.558	0.238	0.436	0.398	0.235	0.487	0.497	0.0
DeltaCon	Cosine	KL	0.021	0.910	0.435	0.902	0.894	0.824	0.438	0.448	0.003
DeltaCon	Cosine	Pearson	0.892	0.905	0.456	0.481	0.448	0.427	0.438	0.54	0.584
DeltaCon	Cosine	Spearman	0.913	0.894	0.462	0.47	0.454	0.409	0.446	0.501	0.860
DeltaCon	Cosine	Xcorr	0.892	0.905	0.57	0.611	0.531	0.557	0.571	0.55	0.797
DeltaCon	Cosine	DTW	0.316	0.859	0.659	0.815	0.792	0.726	0.833	0.841	0.205
DeltaCon	Euclidean	Sym	0.432	0.752	0.802	0.741	0.737	0.706	0.893	0.894	0.232
DeltaCon	Euclidean	Gauss	0.001	0.273	0.519	0.24	0.227	0.165	0.752	0.762	0.0
DeltaCon	Euclidean	KL	0.264	0.859	0.862	0.848	0.843	0.807	0.963	0.965	0.002
DeltaCon	Euclidean	Pearson	0.902	0.907	0.452	0.479	0.448	0.43	0.439	0.54	0.657
DeltaCon	Euclidean	Spearman	0.913	0.895	0.462	0.469	0.452	0.409	0.446	0.501	0.880
DeltaCon	Euclidean	Xcorr	0.902	0.907	0.55	0.612	0.533	0.555	0.571	0.554	0.802
DeltaCon	Euclidean	DTW	0.432	0.752	0.828	0.741	0.737	0.706	0.918	0.926	0.232
Spectral	Cosine	Sym	0.887	0.368	0.17	0.401	0.411	0.476	0.017	0.012	0.988
Spectral	Cosine	Gauss	0.733	0.003	0.040	0.001	0.001	0.005	0.0	0.0	0.990
Spectral	Cosine	KL	0.903	0.154	0.197	0.128	0.151	0.289	0.0	0.0	0.993
Spectral	Cosine	Pearson	0.646	0.667	0.531	0.598	0.463	0.414	0.547	0.437	0.57
Spectral	Cosine	Spearman	0.609	0.567	0.515	0.595	0.453	0.424	0.53	0.419	0.648
Spectral	Cosine	Xcorr	0.646	0.667	0.531	0.598	0.493	0.514	0.547	0.571	0.667
Spectral	Cosine	DTW	0.887	0.368	0.17	0.401	0.411	0.476	0.017	0.012	0.991
Spectral	Euclidean	Sym	0.770	0.45	0.386	0.461	0.466	0.497	0.309	0.307	0.970
Spectral	Euclidean	Gauss	0.339	0.003	0.013	0.002	0.002	0.005	0.0	0.0	0.974
Spectral	Euclidean	KL	0.770	0.232	0.193	0.264	0.274	0.337	0.013	0.009	0.980
Spectral	Euclidean	Pearson	0.661	0.673	0.525	0.597	0.462	0.415	0.545	0.44	0.594
Spectral	Euclidean	Spearman	0.610	0.567	0.516	0.596	0.451	0.424	0.53	0.419	0.609
Spectral	Euclidean	Xcorr	0.661	0.673	0.525	0.597	0.495	0.516	0.545	0.571	0.624
Spectral	Euclidean	DTW	0.770	0.45	0.386	0.461	0.466	0.497	0.309	0.307	0.973
Euclidean	Cosine	Sym	0.546	0.935	0.592	0.907	0.913	0.88	0.584	0.579	0.436
Euclidean	Cosine	Gauss	0.014	0.882	0.262	0.791	0.809	0.706	0.12	0.082	0.004
Euclidean	Cosine	KL	0.387	0.984	0.481	0.975	0.977	0.962	0.27	0.26	0.006
Euclidean	Cosine	Pearson	0.994	1.000	0.501	0.52	0.539	0.362	0.445	0.577	0.629
Euclidean	Cosine	Spearman	0.993	0.991	0.508	0.492	0.516	0.373	0.449	0.569	0.882
Euclidean	Cosine	Xcorr	0.994	1.000	0.512	0.565	0.539	0.586	0.578	0.577	0.793
Euclidean	Cosine	DTW	0.546	0.946	0.648	0.942	0.946	0.938	0.61	0.579	0.436
Euclidean	Euclidean	Sym	0.663	0.966	0.833	0.909	0.909	0.89	0.863	0.872	0.463
Euclidean	Euclidean	Gauss	0.09	0.967	0.56	0.806	0.805	0.736	0.64	0.663	0.007
Euclidean	Euclidean	KL	0.735	0.997	0.905	0.978	0.978	0.967	0.937	0.942	0.036
Euclidean	Euclidean	Pearson	0.999	0.999	0.492	0.517	0.538	0.365	0.442	0.58	0.727
Euclidean	Euclidean	Spearman	0.992	0.991	0.508	0.49	0.515	0.374	0.449	0.569	0.956
Euclidean	Euclidean	Xcorr	0.999	0.999	0.529	0.566	0.538	0.584	0.577	0.58	0.861
Euclidean	Euclidean	DTW	0.663	0.966	0.89	0.919	0.916	0.921	0.904	0.893	0.463

averaged integrity $\bar{I}$; within each row the best, second-best and third-best scores are highlighted in gold, silver and bronze respectively.

4.3.1 Results Across Scenarios

- **Gradual–Merge.** UASE attains the highest integrity ($\bar{I} = 0.996$), confirming its role as a stability oracle; IPP follows with 0.872. Among neural models, the Graph Atten-

Table 4.5: Integrity-gap and win statistics for the chosen index (Euclidean, Euclidean, Pearson) in both not aligned and aligned versions. The first column shows if representation alignment is performed or not.

Alignment	$\Delta\bar{I}_{\text{UASE}}$	$\Delta\bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE\|IPP}}$
no	0.502	0.466	3
yes	0.468	0.431	3

tion Network (GAT) outperforms the temporal models, indicating that slow, smooth drift can be captured without temporal memory.

- **Abrupt–Move.** UASE leads (0.990) and UASE comes a close second (0.966), showing they both react well to a sudden, community-level shift. The simple Auto Encoder based model (AE) (0.552) is the best of the learning-based models, narrowly edging the dynamic variants. Here the models that try to learn temporal dynamics-DynRNN, DynAE, DynAERNN, StConv-do not yet improve on a static auto-encoder, implying they struggle to track a sudden migration.

- **Periodic–Transitions.** IPP and UASE again dominate (0.999 and 0.998). yet now a dynamic architecture finally stands out: StConv reaches 0.727, outscoring every other learned method by a wide margin and claiming the bronze rank overall. The temporal convolution evidently helps when the underlying graph truly follows a rhythmic pattern, whereas static models still plateau near 0.5.

Integrity versus downstream link prediction. To assess whether higher integrity aligns with better task performance we use the Spearman rank correlation, which captures monotone rather than strictly linear associations. For each scenario we correlate the eight in-

Table 4.6: Integrity scores for each model on the three synthetic scenarios, computed with the validated Euclidean–Euclidean–Pearson integrity index. Models are grouped by learning paradigm; the Alignment column indicates whether representation alignment was applied (Yes) or not (No) before the metric was computed.

Alignment	Category	Model	Dataset		
			Merge	*Move*	*Periodic*
No	Non learning-based	IPP	0.872	0.966	0.999
		UASE	0.996	0.990	0.999
	Static	GAT	0.585	0.503	0.492
		AE	0.509	0.552	0.517
		AE_Shuffled	0.530	0.517	0.538
	Dynamic	DynAE	0.527	0.524	0.365
		DynRNN	0.541	0.535	0.442
		DynAERNN	0.520	0.536	0.580
		STConv	0.409	0.516	0.727
Yes	Non learning-based	IPP	0.866	0.890	0.999
		UASE	0.995	0.874	0.998
	Static	GAT	0.45	0.467	0.542
		AE	0.699	0.442	0.961
		AE_Shuffled	0.581	0.512	0.425
	Dynamic	DynAE	0.552	0.404	0.965
		DynRNN	0.522	0.459	0.556
		DynAERNN	0.318	0.627	0.427
		STConv	0.411	0.52	0.738

tegrity scores with the eight one-step link-prediction AUCs of the same models:

$$\rho(\bar{I}, \mathrm{AUC}) = \begin{cases} 0.786 & \text{Gradual-Merge} \\ 0.216 & \text{Abrupt-Move} \\ 0.671 & \text{Periodic} \end{cases} \qquad \text{mean } \rho = 0.596.$$

The rank correlation remains strong for the smooth *Merge* stream and moderate for the *Periodic* stream, but drops to a weak value in the *Move* scenario, where AUC appears less sensitive to integrity differences. On average ($\rho \approx 0.596$) the monotone relation persists:

models that preserve temporal consistency tend-though not uniformly-to achieve higher link-prediction accuracy.

4.3.2 Results in Real Dataset

In this section we compare the integrity scores of graph models trained on a real dataset, using the validated integrity index. For this experiment we use CanParl [57] Dataset, which is a dynamic political network that tracks the interactions of Canadian Members of Parliament (MPs) from 2006 to 2019. Each node represents an MP, and an edge connects them if both voted "yes" on a bill.

The results are shown in table 4.7. The scores show a clear hierarchy: STConv dominates, achieving a near-perfect integrity score (0 .980), well ahead of every other method. The only models in its vicinity are the static auto-encoder (AE, 0 .671) and the non-learning baseline IPP (0 .647), which take silver and bronze, respectively. All remaining dynamic variants (DynAE, DynRNN, DynAERNN) cluster in the mid-0.5 range, indicating that the extra temporal complexity offered no advantage here, while the graph-attention model (GAT, 0 .404) trails far behind. In short, a simple spatio-temporal convolution captures the scenario's structure best, static AE delivers a solid but distant second, and most dynamic or attention-based models contribute little beyond baseline performance.

Table 4.7: Integrity scores in CanParl Dataset.

	IPP	**UASE**	**GAT**	**AE**	**DynAE**	**DynRNN**	**DynAERNN**	**STConv**
Integrity score	0.647	0.616	0.404	0.671	0.617	0.552	0.470	0.980

Chapter 5

Discussions and Conclusion

This thesis set out to answer a fundamental question: *How can we tell whether a dynamic graph–learning model is faithfully capturing the evolving structure of a network, irrespective of any downstream task?* To that end, we introduced the notion of *representation integrity*, formalised a modular design space of integrity indexes, and screened forty-two candidate metrics against a principled set of properties and synthetic tests. The outcome is a single, validated index-Euclidean graph change, Euclidean representation change, Pearson alignment-that we then used to benchmark eight representative embedding models on three synthetic scenarios and a real parliamentary-voting dataset. This chapter interprets those results, situates them in the broader literature, and reflects on the practical and theoretical implications of integrity-oriented evaluation.

Prior work on dynamic embeddings has largely relied on task performance-link prediction, node classification, or anomaly detection-as a proxy for quality. While convenient, such proxies conflate representational faithfulness with task-specific inductive biases. Our first contribution was therefore conceptual: we framed integrity as a task-agnostic, bidirectional coupling between graph dynamics and embedding dynamics. Building on that foundation, we argued that any integrity index must (i) compare graph change to representation change in the *same* time window, and (ii) account for arbitrary rigid motions through an alignment kernel.

The methodological pipeline that followed-property selection, synthetic scenario design, exhaustive metric screening-was motivated by a need for *construct validity*. Only an index that passes all properties and reproduces known ground-truth relationships in controlled settings can meaningfully adjudicate between models on real data.

The exhaustive search revealed that most intuitive combinations of distance measures fail at least one property or scenario. Our chosen index, the Euclidean–Euclidean–Pearson triplet, succeeds because (a) Euclidean distance linearises small perturbations in both the graph and embedding spaces, and (b) Pearson correlation removes scale dependence introduced by alignment.

Model Comparison

Across all scenarios, non-learning based models consistently outperform others, with or without procrustes alignment performed on the representation. Without the alignment, we see that STConv is the only model that achieves a considerably high result. We see that dynamic models unfortunately don't outperform static models generally. However, the alignment procedure helps Auto Encoder based models like AE, DynAE and DynAERNN to achieve much better integrity scores.

Integrity as a Proxy for Task Performance

A strong average Spearman correlation ($\rho = 0.596$) between integrity and one-step link-prediction AUC on the synthetic datasets shows that integrity has a positive correlation with downstream effectiveness.

5.1 Implications

Temporal convolution (or any operation that couples multiple snapshots in the feature space) emerges as a first-class design principle. Architectures that treat time as a mere

sequence of static graphs risk overlooking long-range correlations that integrity penalizes and downstream tasks implicitly reward.

The validated index provides a lightweight, task-independent diagnostic tool. Practitioners can now evaluate candidate models on *their own* data before committing to the heavy engineering effort of task-specific fine-tuning. In practice, integrity scores can guide concrete decisions: models with low integrity may be ruled out early, while models with high integrity can be prioritized for further optimization.

This allows practitioners to allocate computational budgets more effectively, choose architectures that are more likely to remain robust under distribution shifts, and build greater confidence in model stability before deployment.

Our property-based framework opens two research directions: (i) deriving tighter theoretical bounds on integrity for classes of graph processes; and (ii) designing learning objectives that maximize integrity explicitly, analogous to how contrastive losses maximize mutual information.

5.2 Limitations

Three caveats temper our conclusions. First, the synthetic scenarios, while diverse, cannot span the full variability of real-world dynamics; integrity behavior under adversarial or highly non-stationary conditions remains an open question. Second, the benchmark set covers only eight models; emerging techniques such as continuous-time diffusion-based embeddings were beyond our computational budget. Third, we validated a single index, but alternative graph distances or alignment kernels may prove suitable if paired with different properties-a line worth exploring.

5.3 Future Work

Two strands of future research appear particularly promising:

1. **Integrity-aware training.** Incorporating the validated index (or a differentiable surrogate) into the loss function could steer models toward embeddings that are both predictive and faithful.

2. **Domain-specific scenarios.** Crafting stress tests that mimic domain-typical phenomena-e.g. churn in social networks, bursts in communication graphs-would sharpen the diagnostic power of integrity beyond generic synthetic benchmarks.

5.4 Conclusion

By disentangling representational faithfulness from task performance, this thesis provides a principled yardstick-*representation integrity*-for the rapidly evolving field of dynamic graph learning. Our empirical study demonstrates that integrity both discriminates meaningfully among models and correlates with predictive success, making it an actionable criterion for model selection. The broader lesson is that good representations *move* in ways that mirror their data: respecting that motion is not a luxury but a necessity for robust, generalizable learning.

Appendix A

A.1 Index validation - Integrity gap and win statistics for all indexes before and after Procrustes Alignment

Tables A.1 and A.2 present the validation results for all indexes where representations are either used without modification or aligned before the integrity score computation, respectively.

Table A.1: Integrity-gap and win statistics for every index in nonaligned version.

ΔG	ΔR	kernel	$\Delta \bar{I}_{\text{UASE}}$	$\Delta \bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE—IPP}}$
Euclidean	Euclidean	DTW	0.048	-0.237	3
Euclidean	Cosine	DTW	0.086	-0.289	3
Spectral	Euclidean	Spearman	0.003	0.147	3
Spectral	Euclidean	Pearson	0.158	0.264	3
Euclidean	Cosine	Sym	0.088	-0.275	3
Euclidean	Cosine	Gauss	0.305	-0.576	3
Euclidean	Cosine	KL	0.062	-0.617	3
Euclidean	Cosine	Pearson	0.499	0.472	3
Spectral	Cosine	Xcorr	0.121	0.228	3

Continued on next page

Table A.1: Integrity-gap and win statistics (continued).

ΔG	ΔR	kernel	$\Delta \bar{I}_{\text{UASE}}$	$\Delta \bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE—IPP}}$
Spectral	Cosine	Spearman	0.003	0.147	3
Spectral	Cosine	Pearson	0.152	0.263	3
Euclidean	Cosine	Spearman	0.548	0.271	3
Euclidean	Cosine	Xcorr	0.435	0.424	3
Euclidean	Euclidean	Sym	0.051	-0.231	3
DeltaCon	Euclidean	Xcorr	0.112	0.281	3
DeltaCon	Euclidean	Spearman	0.034	0.280	3
DeltaCon	Euclidean	Pearson	0.152	0.332	3
Euclidean	Euclidean	Gauss	0.132	-0.693	3
Euclidean	Euclidean	KL	0.016	-0.243	3
Euclidean	Euclidean	Pearson	0.502	0.466	3
Euclidean	Euclidean	Spearman	0.548	0.273	3
DeltaCon	Cosine	Xcorr	0.106	0.272	3
DeltaCon	Cosine	Spearman	0.033	0.282	3
DeltaCon	Cosine	Pearson	0.146	0.323	3
Euclidean	Euclidean	Xcorr	0.438	0.418	3
Spectral	Euclidean	Xcorr	0.129	0.229	3
Spectral	Euclidean	DTW	-0.037	0.281	2
DeltaCon	Cosine	Sym	0.049	-0.392	2
Spectral	Euclidean	KL	-0.054	0.441	2
Spectral	Euclidean	Gauss	-0.059	0.454	2
DeltaCon	Cosine	Gauss	-0.007	-0.360	2
Spectral	Cosine	DTW	-0.049	0.392	2

Continued on next page

Table A.1: Integrity-gap and win statistics (continued).

ΔG	ΔR	kernel	$\Delta\bar{I}_{\text{UASE}}$	$\Delta\bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE—IPP}}$
Spectral	Cosine	KL	-0.045	0.589	2
Spectral	Cosine	Gauss	-0.170	0.666	2
Spectral	Cosine	Sym	-0.048	0.393	2
DeltaCon	Cosine	DTW	0.046	-0.407	2
DeltaCon	Cosine	KL	0.156	-0.610	2
Spectral	Euclidean	Sym	-0.037	0.281	2
DeltaCon	Euclidean	DTW	0.036	-0.282	1
DeltaCon	Euclidean	KL	0.086	-0.578	1
DeltaCon	Euclidean	Gauss	-0.007	-0.215	1
DeltaCon	Euclidean	Sym	0.037	-0.281	1

Table A.2: Integrity-gap and win statistics for every index in aligned version.

ΔG	ΔR	kernel	$\Delta\bar{I}_{\text{UASE}}$	$\Delta\bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE—IPP}}$
DeltaCon	Cosine	Sym	0.214	-0.099	3
DeltaCon	Cosine	Gauss	0.029	0.000	3
Euclidean	Euclidean	Xcorr	0.435	0.414	3
Euclidean	Euclidean	Spearman	0.548	0.271	3
Euclidean	Euclidean	Pearson	0.468	0.431	3
Euclidean	Euclidean	KL	0.117	-0.189	3
Euclidean	Euclidean	Gauss	0.382	-0.294	3
Euclidean	Euclidean	Sym	0.130	-0.117	3
Euclidean	Cosine	DTW	0.223	-0.100	3

Continued on next page

Table A.2: Integrity-gap and win statistics (continued).

ΔG	ΔR	kernel	$\Delta \bar{I}_{\text{UASE}}$	$\Delta \bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE—IPP}}$
Euclidean	Cosine	Xcorr	0.435	0.424	3
Euclidean	Cosine	Spearman	0.548	0.269	3
Euclidean	Cosine	Pearson	0.470	0.442	3
Euclidean	Cosine	KL	0.317	-0.450	3
Euclidean	Cosine	Gauss	0.516	-0.123	3
Euclidean	Cosine	Sym	0.213	-0.100	3
Euclidean	Euclidean	DTW	0.141	-0.117	3
DeltaCon	Euclidean	DTW	0.154	-0.116	3
DeltaCon	Euclidean	Gauss	0.056	-0.004	3
DeltaCon	Euclidean	Sym	0.154	-0.116	3
DeltaCon	Cosine	DTW	0.214	-0.099	3
DeltaCon	Euclidean	KL	0.310	-0.345	3
DeltaCon	Cosine	KL	0.554	-0.045	3
DeltaCon	Cosine	Pearson	0.157	0.331	2
DeltaCon	Cosine	Spearman	0.047	0.290	2
DeltaCon	Cosine	Xcorr	0.111	0.276	2
DeltaCon	Euclidean	Xcorr	0.116	0.284	2
DeltaCon	Euclidean	Spearman	0.048	0.287	2
DeltaCon	Euclidean	Pearson	0.161	0.338	2
Spectral	Euclidean	Pearson	0.179	0.284	1
Spectral	Euclidean	Xcorr	0.102	0.201	1
Spectral	Cosine	Xcorr	0.092	0.197	1
Spectral	Cosine	Spearman	0.025	0.157	1

Continued on next page

Table A.2: Integrity-gap and win statistics (continued).

ΔG	ΔR	kernel	$\Delta \bar{I}_{\text{UASE}}$	$\Delta \bar{I}_{\text{IPP}}$	$\text{Win}_{\text{UASE—IPP}}$
Spectral	Cosine	Pearson	0.176	0.287	1
Spectral	Euclidean	Spearman	0.022	0.155	1
Spectral	Cosine	KL	-0.288	0.106	0
Spectral	Cosine	Sym	-0.213	0.099	0
Spectral	Euclidean	DTW	-0.153	0.117	0
Spectral	Euclidean	KL	-0.233	0.145	0
Spectral	Euclidean	Gauss	-0.197	0.399	0
Spectral	Cosine	Gauss	-0.546	0.316	0
Spectral	Cosine	DTW	-0.214	0.099	0
Spectral	Euclidean	Sym	-0.153	0.117	0

A.2 Integrity Scores after Procrustes Alignment

After performing Procrustes alignment on the representations, we compute integrity scores and report them in Tables A.3, A.4, and A.5 for merge, move, and periodic settings, respectively.

In Table A.6, we present the integrity scores of various models based on the validated integrity index - Euclidean, Euclidean and Pearson composition. In the merge setting, UASE achieves the highest integrity score of 0.995, slightly ahead of IPP. All other learned models fall significantly behind, with AE scoring 0.699, the best among the others. In the move setting, IPP (0.890) surpasses UASE, while the only learned model that narrows the gap is DynAERNN (0.627), indicating that its recurrent attention mechanism effectively tracks the shifting community. In the periodic setting, IPP narrowly outperforms UASE (0.999 versus 0.998), but DynAE stands out among the neural approaches with a score of 0.965.

Table A.3: Integrity scores in merge setting, when representations are aligned between timestamps.

			Non-learning-based		Static			Dynamic			
ΔG	ΔR	kernel	IPP	UASE	GAT	AE	AE-Shuffled	DynAE	DynRNN	DynAERNN	STConv
DeltaCon	Cosine	Sym	0.267	0.601	0.249	0.338	0.357	0.253	0.228	0.228	0.233
DeltaCon	Cosine	Gauss	0.0	0.041	0.0	0.000	0.000	0.0	0.0	0.0	0.0
DeltaCon	Cosine	KL	0.0	0.659	0.005	0.006	0.037	0.0	0.0	0.0	0.0
DeltaCon	Cosine	Pearson	0.798	0.637	0.471	0.847	0.581	0.799	0.5	0.644	0.605
DeltaCon	Cosine	Spearman	0.74	0.586	0.459	0.807	0.575	0.792	0.5	0.770	0.749
DeltaCon	Cosine	Xcorr	0.807	0.674	0.495	0.853	0.613	0.802	0.59	0.66	0.755
DeltaCon	Cosine	DTW	0.267	0.601	0.249	0.338	0.357	0.253	0.228	0.228	0.233
DeltaCon	Euclidean	Sym	0.365	0.658	0.282	0.454	0.477	0.336	0.228	0.238	0.241
DeltaCon	Euclidean	Gauss	0.0	0.091	0.0	0.002	0.004	0.0	0.0	0.0	0.0
DeltaCon	Euclidean	KL	0.034	0.749	0.025	0.313	0.387	0.001	0.0	0.0	0.0
DeltaCon	Euclidean	Pearson	0.803	0.638	0.46	0.849	0.579	0.813	0.509	0.697	0.623
DeltaCon	Euclidean	Spearman	0.739	0.587	0.476	0.809	0.576	0.792	0.581	0.77	0.794
DeltaCon	Euclidean	Xcorr	0.811	0.676	0.472	0.853	0.612	0.813	0.595	0.721	0.806
DeltaCon	Euclidean	DTW	0.365	0.658	0.282	0.454	0.477	0.336	0.228	0.238	0.241
Spectral	Cosine	Sym	0.963	0.628	0.979	0.891	0.872	0.977	0.999	0.999	0.996
Spectral	Cosine	Gauss	0.967	0.056	0.967	0.75	0.691	0.987	1.000	1.000	0.994
Spectral	Cosine	KL	0.966	0.54	0.98	0.889	0.869	0.98	0.985	0.998	0.997
Spectral	Cosine	Pearson	0.660	0.498	0.46	0.752	0.535	0.754	0.5	0.6	0.575
Spectral	Cosine	Spearman	0.652	0.536	0.476	0.750	0.553	0.732	0.5	0.699	0.698
Spectral	Cosine	Xcorr	0.669	0.523	0.478	0.752	0.608	0.758	0.571	0.778	0.693
Spectral	Cosine	DTW	0.963	0.628	0.98	0.891	0.872	0.977	0.999	0.999	0.997
Spectral	Euclidean	Sym	0.864	0.571	0.947	0.776	0.752	0.893	0.999	0.991	0.988
Spectral	Euclidean	Gauss	0.661	0.019	0.889	0.346	0.263	0.773	1.000	0.996	0.992
Spectral	Euclidean	KL	0.859	0.447	0.947	0.75	0.721	0.893	0.987	0.993	0.991
Spectral	Euclidean	Pearson	0.665	0.5	0.455	0.747	0.536	0.753	0.488	0.642	0.585
Spectral	Euclidean	Spearman	0.652	0.535	0.492	0.752	0.555	0.731	0.54	0.698	0.711
Spectral	Euclidean	Xcorr	0.673	0.526	0.476	0.747	0.612	0.755	0.579	0.810	0.716
Spectral	Euclidean	DTW	0.864	0.571	0.947	0.776	0.752	0.893	0.999	0.992	0.988
Euclidean	Cosine	Sym	0.506	0.840	0.488	0.577	0.596	0.491	0.467	0.467	0.472
Euclidean	Cosine	Gauss	0.006	0.567	0.007	0.029	0.037	0.006	0.003	0.003	0.007
Euclidean	Cosine	KL	0.023	0.947	0.068	0.406	0.518	0.016	0.0	0.0	0.009
Euclidean	Cosine	Pearson	0.865	0.996	0.482	0.667	0.595	0.506	0.5	0.301	0.396
Euclidean	Cosine	Spearman	0.849	0.992	0.411	0.562	0.663	0.516	0.5	0.471	0.6
Euclidean	Cosine	Xcorr	0.918	0.998	0.507	0.728	0.607	0.533	0.546	0.337	0.586
Euclidean	Cosine	DTW	0.506	0.840	0.488	0.577	0.596	0.491	0.467	0.467	0.472
Euclidean	Euclidean	Sym	0.604	0.897	0.521	0.692	0.716	0.575	0.467	0.477	0.48
Euclidean	Euclidean	Gauss	0.037	0.783	0.019	0.150	0.192	0.027	0.003	0.005	0.007
Euclidean	Euclidean	KL	0.56	0.977	0.18	0.760	0.808	0.443	0.0	0.013	0.009
Euclidean	Euclidean	Pearson	0.866	0.995	0.45	0.699	0.581	0.552	0.522	0.318	0.411
Euclidean	Euclidean	Spearman	0.849	0.992	0.423	0.562	0.661	0.516	0.388	0.474	0.654
Euclidean	Euclidean	Xcorr	0.919	0.998	0.475	0.754	0.594	0.575	0.533	0.367	0.67
Euclidean	Euclidean	DTW	0.604	0.912	0.521	0.692	0.716	0.575	0.467	0.477	0.48

When comparing integrity scores before alignment in Table 4.6, IPP and UASE consistently rank as the top two models. However, the scores for AE and DynAE show a significant improvement, whereas the scores for other models do not show as drastic a boost.

Table A.4: Integrity scores in move setting.

			Non-learning-based		Static			Dynamic			
ΔG	ΔR	kernel	IPP	UASE	GAT	AE	AE-Shuffled	DynAE	DynRNN	DynAERNN	STConv
DeltaCon	Cosine	Sym	0.236	0.516	0.249	0.345	0.348	0.272	0.204	0.207	0.217
DeltaCon	Cosine	Gauss	0.0	0.006	0.002	0.0	0.000	0.0	0.0	0.0	0.0
DeltaCon	Cosine	KL	0.0	0.499	0.038	0.001	0.013	0.0	0.0	0.0	0.004
DeltaCon	Cosine	Pearson	0.884	0.506	0.474	0.775	0.416	0.740	0.5	0.673	0.546
DeltaCon	Cosine	Spearman	0.815	0.246	0.48	0.780	0.399	0.794	0.5	0.592	0.6
DeltaCon	Cosine	Xcorr	0.884	0.506	0.537	0.775	0.52	0.766	0.614	0.673	0.644
DeltaCon	Cosine	DTW	0.236	0.516	0.249	0.345	0.348	0.272	0.204	0.207	0.217
DeltaCon	Euclidean	Sym	0.329	0.597	0.288	0.466	0.470	0.386	0.204	0.239	0.228
DeltaCon	Euclidean	Gauss	0.0	0.030	0.002	0.002	0.002	0.0	0.0	0.0	0.0
DeltaCon	Euclidean	KL	0.0	0.657	0.077	0.376	0.386	0.112	0.0	0.0	0.003
DeltaCon	Euclidean	Pearson	0.895	0.52	0.464	0.779	0.416	0.745	0.47	0.644	0.584
DeltaCon	Euclidean	Spearman	0.813	0.252	0.485	0.780	0.396	0.795	0.455	0.59	0.75
DeltaCon	Euclidean	Xcorr	0.895	0.52	0.54	0.779	0.516	0.770	0.599	0.644	0.668
DeltaCon	Euclidean	DTW	0.329	0.597	0.288	0.466	0.470	0.386	0.204	0.239	0.228
Spectral	Cosine	Sym	0.969	0.69	0.954	0.861	0.858	0.933	0.998	0.997	0.987
Spectral	Cosine	Gauss	0.979	0.119	0.9	0.651	0.636	0.902	0.997	0.998	0.987
Spectral	Cosine	KL	0.973	0.636	0.945	0.856	0.852	0.935	0.978	0.997	0.989
Spectral	Cosine	Pearson	0.975	0.781	0.476	0.572	0.487	0.523	0.5	0.729	0.514
Spectral	Cosine	Spearman	0.664	0.395	0.482	0.646	0.479	0.645	0.5	0.532	0.557
Spectral	Cosine	Xcorr	0.975	0.781	0.564	0.572	0.575	0.551	0.649	0.729	0.578
Spectral	Cosine	DTW	0.969	0.69	0.96	0.862	0.859	0.933	0.998	0.997	0.987
Spectral	Euclidean	Sym	0.877	0.609	0.916	0.74	0.736	0.82	0.998	0.966	0.977
Spectral	Euclidean	Gauss	0.714	0.034	0.801	0.225	0.215	0.488	0.997	0.970	0.982
Spectral	Euclidean	KL	0.874	0.512	0.907	0.706	0.7	0.807	0.982	0.970	0.980
Spectral	Euclidean	Pearson	0.956	0.791	0.468	0.57	0.488	0.526	0.473	0.647	0.552
Spectral	Euclidean	Spearman	0.664	0.395	0.485	0.647	0.478	0.643	0.511	0.537	0.633
Spectral	Euclidean	Xcorr	0.956	0.791	0.591	0.57	0.571	0.552	0.638	0.647	0.552
Spectral	Euclidean	DTW	0.877	0.609	0.922	0.74	0.736	0.82	0.998	0.966	0.977
Euclidean	Cosine	Sym	0.672	0.953	0.682	0.782	0.785	0.709	0.64	0.644	0.654
Euclidean	Cosine	Gauss	0.095	0.948	0.151	0.354	0.364	0.159	0.059	0.062	0.078
Euclidean	Cosine	KL	0.393	0.994	0.234	0.848	0.855	0.637	0.0	0.002	0.041
Euclidean	Cosine	Pearson	0.919	0.870	0.468	0.445	0.51	0.402	0.5	0.716	0.491
Euclidean	Cosine	Spearman	0.23	0.929	0.499	0.146	0.577	0.123	0.500	0.46	0.391
Euclidean	Cosine	Xcorr	0.919	0.870	0.562	0.445	0.592	0.43	0.631	0.716	0.53
Euclidean	Cosine	DTW	0.672	0.953	0.682	0.782	0.785	0.709	0.64	0.644	0.654
Euclidean	Euclidean	Sym	0.766	0.962	0.72	0.903	0.907	0.823	0.64	0.676	0.665
Euclidean	Euclidean	Gauss	0.3	0.965	0.245	0.808	0.823	0.505	0.059	0.101	0.089
Euclidean	Euclidean	KL	0.82	0.996	0.431	0.975	0.977	0.909	0.0	0.386	0.272
Euclidean	Euclidean	Pearson	0.890	0.874	0.467	0.442	0.512	0.404	0.459	0.627	0.52
Euclidean	Euclidean	Spearman	0.232	0.924	0.48	0.148	0.579	0.121	0.488	0.454	0.189
Euclidean	Euclidean	Xcorr	0.890	0.874	0.596	0.442	0.589	0.43	0.636	0.627	0.52
Euclidean	Euclidean	DTW	0.766	0.962	0.72	0.903	0.907	0.823	0.64	0.676	0.665

Table A.5: Integrity scores in periodic setting.

			Non-learning-based		Static			Dynamic			
ΔG	ΔR	kernel	IPP	UASE	GAT	AE	AE-Shuffled	DynAE	DynRNN	DynAERNN	STConv
DeltaCon	Cosine	Sym	0.272	0.596	0.212	0.362	0.371	0.269	0.189	0.196	0.205
DeltaCon	Cosine	Gauss	0.0	0.041	0.000	0.0	0.000	0.0	0.0	0.0	0.0
DeltaCon	Cosine	KL	0.0	0.641	0.012	0.034	0.086	0.0	0.0	0.0	0.003
DeltaCon	Cosine	Pearson	0.896	0.911	0.541	0.883	0.408	0.885	0.5	0.383	0.588
DeltaCon	Cosine	Spearman	0.919	0.915	0.556	0.906	0.441	0.902	0.5	0.538	0.859
DeltaCon	Cosine	Xcorr	0.896	0.911	0.63	0.883	0.561	0.885	0.599	0.517	0.795
DeltaCon	Cosine	DTW	0.272	0.596	0.212	0.362	0.371	0.269	0.189	0.196	0.205
DeltaCon	Euclidean	Sym	0.386	0.636	0.242	0.480	0.488	0.385	0.189	0.235	0.234
DeltaCon	Euclidean	Gauss	0.0	0.057	0.0	0.003	0.004	0.0	0.0	0.0	0.0
DeltaCon	Euclidean	KL	0.122	0.716	0.03	0.417	0.433	0.115	0.0	0.003	0.001
DeltaCon	Euclidean	Pearson	0.905	0.913	0.524	0.887	0.412	0.891	0.52	0.433	0.668
DeltaCon	Euclidean	Spearman	0.919	0.915	0.579	0.905	0.441	0.902	0.516	0.531	0.878
DeltaCon	Euclidean	Xcorr	0.905	0.913	0.617	0.887	0.562	0.891	0.625	0.534	0.783
DeltaCon	Euclidean	DTW	0.386	0.636	0.242	0.480	0.488	0.385	0.189	0.235	0.234
Spectral	Cosine	Sym	0.931	0.607	0.971	0.841	0.832	0.934	0.987	0.987	0.989
Spectral	Cosine	Gauss	0.878	0.063	0.955	0.57	0.536	0.901	0.993	0.993	0.990
Spectral	Cosine	KL	0.947	0.53	0.963	0.855	0.844	0.951	0.822	0.980	0.994
Spectral	Cosine	Pearson	0.655	0.680	0.601	0.649	0.439	0.697	0.5	0.429	0.572
Spectral	Cosine	Spearman	0.617	0.606	0.513	0.595	0.449	0.631	0.5	0.472	0.645
Spectral	Cosine	Xcorr	0.655	0.680	0.601	0.649	0.493	0.697	0.556	0.473	0.667
Spectral	Cosine	DTW	0.933	0.607	0.978	0.841	0.832	0.934	0.987	0.993	0.991
Spectral	Euclidean	Sym	0.817	0.567	0.955	0.722	0.715	0.818	0.987	0.965	0.968
Spectral	Euclidean	Gauss	0.486	0.021	0.904	0.182	0.169	0.481	0.993	0.953	0.972
Spectral	Euclidean	KL	0.827	0.468	0.961	0.71	0.699	0.829	0.843	0.973	0.979
Spectral	Euclidean	Pearson	0.669	0.685	0.563	0.654	0.441	0.701	0.541	0.443	0.6
Spectral	Euclidean	Spearman	0.616	0.603	0.528	0.595	0.448	0.631	0.499	0.465	0.606
Spectral	Euclidean	Xcorr	0.669	0.685	0.563	0.654	0.492	0.701	0.572	0.453	0.626
Spectral	Euclidean	DTW	0.817	0.567	0.96	0.722	0.715	0.818	0.987	0.975	0.971
Euclidean	Cosine	Sym	0.502	0.826	0.443	0.592	0.601	0.499	0.42	0.426	0.436
Euclidean	Cosine	Gauss	0.008	0.511	0.011	0.050	0.078	0.01	0.003	0.004	0.004
Euclidean	Cosine	KL	0.158	0.936	0.059	0.569	0.579	0.168	0.0	0.012	0.006
Euclidean	Cosine	Pearson	0.996	0.999	0.552	0.959	0.42	0.961	0.5	0.377	0.633
Euclidean	Cosine	Spearman	0.987	0.983	0.603	0.960	0.444	0.955	0.5	0.524	0.88
Euclidean	Cosine	Xcorr	0.996	0.999	0.636	0.959	0.562	0.961	0.609	0.543	0.791
Euclidean	Cosine	DTW	0.502	0.854	0.447	0.592	0.601	0.499	0.42	0.426	0.436
Euclidean	Euclidean	Sym	0.616	0.866	0.473	0.711	0.718	0.615	0.42	0.466	0.465
Euclidean	Euclidean	Gauss	0.049	0.664	0.022	0.216	0.259	0.062	0.003	0.013	0.007
Euclidean	Euclidean	KL	0.635	0.958	0.111	0.806	0.808	0.633	0.0	0.089	0.046
Euclidean	Euclidean	Pearson	0.999	0.998	0.542	0.961	0.425	0.965	0.556	0.427	0.738
Euclidean	Euclidean	Spearman	0.987	0.983	0.634	0.960	0.443	0.955	0.573	0.517	0.95
Euclidean	Euclidean	Xcorr	0.999	0.998	0.633	0.961	0.561	0.965	0.637	0.529	0.842
Euclidean	Euclidean	DTW	0.616	0.886	0.473	0.711	0.718	0.615	0.42	0.466	0.465

Table A.6: Integrity scores for each model after representation alignment.

Category	**Model**	**Dataset**		
		Merge	*Move*	*Periodic*
Non learning-based	IPP	0.866	0.890	0.999
	UASE	0.995	0.874	0.998
Static	GAT	0.450	0.467	0.542
	AE	0.699	0.442	0.961
	AE-Shuffled	0.581	0.512	0.425
Dynamic	DynAE	0.552	0.404	0.965
	DynRNN	0.522	0.459	0.556
	DynAERNN	0.318	0.627	0.427
	STConv	0.411	0.520	0.738

Bibliography

[1] ABU-EL-HAIJA, S., PEROZZI, B., AL-RFOU, R., AND ALEMI, A. Watch your step: learning node embeddings via graph attention. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2018), NIPS'18, Curran Associates Inc., p. 9198–9208.

[2] AGARWAL, C., LAKKARAJU, H., AND ZITNIK, M. Towards a unified framework for fair and stable graph representation learning, 2021.

[3] AL-RFOU, R., PEROZZI, B., AND ZELLE, D. Ddgk: Learning graph representations for deep divergence graph kernels. In *The World Wide Web Conference* (New York, NY, USA, 2019), WWW '19, Association for Computing Machinery, p. 37–48.

[4] AL-SAYOURI, S. A., KOUTRA, D., PAPALEXAKIS, E. E., AND LAM, S. S. Recs: Robust graph embedding using connection subgraphs, 2018.

[5] ANTONIAK, M., AND MIMNO, D. Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics 6* (2018), 107–119.

[6] BARROS, C. D. T., MENDONÇA, M. R. F., VIEIRA, A. B., AND ZIVIANI, A. A survey on embedding dynamic graphs.

[7] BECHLER-SPEICHER, M., FINKELSHTEIN, B., FRASCA, F., MÜLLER, L., TÖNSHOFF, J., SIRAUDIN, A., ZAVERKIN, V., BRONSTEIN, M. M., NIEPERT, M., PEROZZI, B.,

GALKIN, M., AND MORRIS, C. Position: Graph learning will lose relevance due to poor benchmarks, 2025.

[8] BELKIN, M., AND NIYOGI, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems* (2001).

[9] BELKIN, M., AND NIYOGI, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15* (2003), 1373–1396.

[10] BHAGAT, S., CORMODE, G., AND MUTHUKRISHNAN, S. Node Classification in Social Networks. In *Social Network Data Analytics*, C. C. Aggarwal, Ed. 2011, p. 115.

[11] BO, D., WANG, X., SHI, C., ZHU, M., LU, E., AND CUI, P. Structural deep clustering network. *Proceedings of The Web Conference 2020* (2020).

[12] BORAH, A., BARMAN, M. P., AND AWEKAR, A. Are word embedding methods stable and should we care about it? In *Proceedings of the 32nd ACM Conference on Hypertext and Social Media* (New York, NY, USA, 2021), HT '21, Association for Computing Machinery, p. 45–55.

[13] CHEN, H., AND LI, J. Exploiting structural and temporal evolution in dynamic link prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2018), CIKM '18, Association for Computing Machinery, p. 427–436.

[14] CUI, P., WANG, X., PEI, J., AND ZHU, W. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering 31*, 5 (2019), 833–852.

[15] DA SILVA SOARES, P. R., AND PRUDÊNCIO, R. B. C. Time series based link prediction. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (2012), pp. 1–7.

[16] DAVIS, E., GALLAGHER, I., LAWSON, D. J., AND RUBIN-DELANCHY, P. A simple and powerful framework for stable dynamic network embedding.

[17] DILEO, M., ZIGNANI, M., AND GAITO, S. Temporal graph learning for dynamic link prediction with text in online social networks. *Mach. Learn. 113*, 4 (nov 2023), 2207–2226.

[18] DRIDI, A., GABER, M., AZAD, R., AND BHOGAL, J. Vec2dynamics: A temporal word embedding approach to exploring the dynamics of scientific keywords—machine learning as a case study. *Big Data and Cognitive Computing 6*, 1 (Mar. 2022). Publisher Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland.

[19] FALOUTSOS, C., KOUTRA, D., AND VOGELSTEIN, J. T. Deltacon: A principled massive-graph similarity function. In *SDM* (2013).

[20] GALLAGHER, I., JONES, A., AND RUBIN-DELANCHY, P. Spectral embedding for dynamic networks with stability guarantees. In *Proceedings of the 35th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2024), NIPS '21, Curran Associates Inc.

[21] GAO, H., AND HUANG, H. Deep attributed network embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), IJCAI'18, AAAI Press, p. 3364–3370.

[22] GONG, H., BHAT, S., AND VISWANATH, P. Enriching word embeddings with temporal and spatial information. In *Proceedings of the 24th Conference on Computational Natural Language Learning* (Online, Nov. 2020), R. Fernández and T. Linzen, Eds., Association for Computational Linguistics, pp. 1–11.

[23] GOYAL, P., CHHETRI, S. R., AND CANEDO, A. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *ArXiv abs/1809.02657* (2018).

[24] GOYAL, P., KAMRA, N., HE, X., AND LIU, Y. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273* (2018).

[25] GRANDJEAN, M. A social network analysis of twitter: Mapping the digital humanities community. *Cogent Arts & Humanities 3*, 1 (2016), 1171458.

[26] GROVER, A., AND LESKOVEC, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), KDD '16, Association for Computing Machinery, p. 855–864.

[27] GÜRSOY, F., AND BADUR, B. Investigating internal migration with network analysis and latent space representations: an application to turkey. *Soc. Netw. Anal. Min. 12*, 1 (2022), 150.

[28] GUZZI, P. H., AND ROY, S. *Biological network analysis : trends, approaches, graphical theory and algorithms*. Elsevier, Amsterdam, Netherlands ;, 2020.

[29] GÜRSOY, F., HADDAD, M., AND BOTHOREL, C. Alignment and stability of embeddings: measurement and inference improvement, 2021.

[30] HAMILTON, W. L., YING, R., AND LESKOVEC, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS'17, Curran Associates Inc., p. 1025–1035.

[31] HAMILTON, W. L., YING, R., AND LESKOVEC, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull. 40* (2017), 52–74.

[32] HELLRICH, J., KAMPE, B., AND HAHN, U. The influence of down-sampling strategies on SVD word embedding stability. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP* (Minneapolis, USA, June 2019), Association for Computational Linguistics, pp. 18–26.

[33] HOLLAND, P., LASKEY, K. B., AND LEINHARDT, S. Stochastic blockmodels: First steps. *Social Networks 5* (1983), 109–137.

[34] HOLME, P., AND SARAMAKI, J. Temporal networks. *PHYSICS REPORTS: REVIEW SECTION OF PHYSICS LETTERS 519*, 3 (2012), 97–125.

[35] HOLME, P., AND SARAMÄKI, J. Temporal networks. *Physics Reports 519*, 3 (Oct. 2012), 97–125.

[36] JONES, A., AND RUBIN-DELANCHY, P. The multilayer random dot product graph. *ArXiv abs/2007.10455* (2020).

[37] KANG, Z., XU, H., HU, J., AND PEI, X. Learning dynamic graph embedding for traffic flow forecasting: A graph self-attentive method. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (2019), pp. 2570–2576.

[38] KAZEMI, S. M., GOEL, R., JAIN, K., KOBYZEV, I., SETHI, A., FORSYTH, P., AND POUPART, P. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research 21*, 70 (2020), 1–73.

[39] KHOSHRAFTAR, S., AND AN, A. A survey on graph representation learning methods. *ACM Trans. Intell. Syst. Technol. 15*, 1 (jan 2024).

[40] KIM, H., LEE, B. S., SHIN, W.-Y., AND LIM, S. Graph Anomaly Detection With Graph Neural Networks: Current Status and Challenges. *IEEE Access 10* (Jan. 2022), 111820–111829.

[41] KIPF, T., AND WELLING, M. Variational graph auto-encoders. *ArXiv abs/1611.07308* (2016).

[42] KIPF, T. N., AND WELLING, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations* (2017).

[43] KLABUNDE, M., AND LEMMERICH, F. On the prediction instability of graph neural networks, 2022.

[44] KOUTRA, D., SHAH, N., VOGELSTEIN, J. T., GALLAGHER, B., AND FALOUTSOS, C. Deltacon: Principled massive-graph similarity function with attribution. *ACM Trans. Knowl. Discov. Data 10*, 3 (Feb. 2016).

[45] KRZYWDA, M., ŁUKASIK, S., AND GANDOMI, A. H. Graph neural networks in computer vision - architectures, datasets and common approaches. In *2022 International Joint Conference on Neural Networks (IJCNN)* (2022), pp. 1–10.

[46] LANDESBERGER, T. V., KUIJPER, A., SCHRECK, T., KOHLHAMMER, J., WIJK, J. V., FEKETE, J., AND FELLNER, D. W. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum* (2011).

[47] LI, B., AND NABAVI, S. A multimodal graph neural network framework for cancer molecular subtype classification. *BMC Bioinformatics 25* (2023).

[48] LI, H., ZHAO, Y., MAO, Z., QIN, Y., XIAO, Z., FENG, J., GU, Y., JU, W., LUO, X., AND ZHANG, M. Graph neural networks in intelligent transportation systems: Advances, applications and trends.

[49] LI, Y., ZHOU, X., AND PAN, M. *Graph Neural Networks in Urban Intelligence*. Springer Nature Singapore, Singapore, 2022, pp. 579–593.

[50] MAHDAVI, S., KHOSHRAFTAR, S., AND AN, A. Dynamic joint variational graph autoencoders. *ArXiv abs/1910.01963* (2019).

[51] MANESSI, F., ROZZA, A., AND MANZO, M. Dynamic graph convolutional networks. *Pattern Recognition 97* (Jan. 2020), 107000.

[52] MODELL, A., GALLAGHER, I., CECCHERINI, E., WHITELEY, N., AND RUBIN-DELANCHY, P. Intensity profile projection: A framework for continuous-time representation learning for dynamic networks. In *Thirty-seventh Conference on Neural Information Processing Systems* (2023).

[53] NEWMAN, M. *Networks: An Introduction*. Oxford University Press, 03 2010.

[54] NG, A., JORDAN, M., AND WEISS, Y. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 - Proceedings of the 2001 Conference, NIPS 2001* (2002), Advances in Neural Information Processing Systems, Neural information processing systems foundation. 15th Annual Neural Information Processing Systems Conference, NIPS 2001 ; Conference date: 03-12-2001 Through 08-12-2001.

[55] PENSKY, M., AND ZHANG, T. Spectral clustering in the dynamic stochastic block model, 2017.

[56] PEROZZI, B., AL-RFOU, R., AND SKIENA, S. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2014), KDD '14, Association for Computing Machinery, p. 701–710.

[57] POURSAFAEI, F., HUANG, S., PELRINE, K., AND RABBANY, R. Towards better evaluation for dynamic link prediction. *ArXiv abs/2207.10128* (2022).

[58] QIAN, H., ZHOU, H., ZHAO, Q., CHEN, H., YAO, H., WANG, J., LIU, Z., YU, F., ZHANG, Z., AND ZHOU, J. Mdgnn: Multi-relational dynamic graph neural network for comprehensive and dynamic stock investment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2024), vol. 38, pp. 14642–14650.

[59] QUAN, R., WANG, W., MA, F., FAN, H., AND YANG, Y. Clustering for protein representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 319–329.

[60] ROWEIS, S. T., AND SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[61] SCARSELLI, F., GORI, M., TSOI, A. C., HAGENBUCHNER, M., AND MONFARDINI, G. The graph neural network model. *IEEE Transactions on Neural Networks 20*, 1 (2009), 61–80.

[62] SCHUMACHER, T., WOLF, H., RITZERT, M., LEMMERICH, F., BACHMANN, J., FRANTZEN, F., KLABUNDE, M., GROHE, M., AND STROHMAIER, M. The effects of randomness on the stability of node embeddings, 2020.

[63] SHI, B., MORRIS, E., SHEN, H., DU, W., AND SAJJAD, M. H. Geometric instability of graph neural networks on large graphs. In *The Second Learning on Graphs Conference* (2023).

[64] SKARDING, J., GABRYS, B., AND MUSIAL, K. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access 9* (2021), 79143–79168.

[65] TANG, H., WU, S., XU, G., AND LI, Q. Dynamic graph evolution learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2023), SIGIR '23, Association for Computing Machinery, p. 1589–1598.

[66] TANTARDINI, M., IEVA, F., TAJOLI, L., AND PICCARDI, C. Comparing methods for comparing networks. *Scientific Reports 9* (2019).

[67] TSITSULIN, A., PALOWITCH, J., AND PEROZZI, B. Graph clustering with graph neural networks.

[68] VELIČKOVIĆ, P., CUCURULL, G., CASANOVA, A., ROMERO, A., LIÒ, P., AND BENGIO, Y. Graph attention networks. In *International Conference on Learning Representations* (2018).

[69] WANG, C., RAO, W., GUO, W., WANG, P., LIU, J., AND GUAN, X. Towards understanding the instability of network embedding. *IEEE Transactions on Knowledge and Data Engineering 34*, 2 (2022), 927–941.

[70] WANG, D., CUI, P., AND ZHU, W. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and*

Data Mining (New York, NY, USA, 2016), KDD '16, Association for Computing Machinery, p. 1225–1234.

[71] WANG, X., CHENG, N., FU, L., QUAN, W., SUN, R., HUI, Y., LUAN, T., AND SHEN, X. Scalable resource management for dynamic mec: An unsupervised link-output graph neural network approach, 2023.

[72] WEST, D. B. *Introduction to Graph Theory*, second ed. Prentice Hall, Upper Saddle River, N.J., 2001.

[73] XU, K., HU, W., LESKOVEC, J., AND JEGELKA, S. How powerful are graph neural networks? *ArXiv abs/1810.00826* (2018).

[74] XUE, G., ZHONG, M., LI, J., CHEN, J., ZHAI, C., AND KONG, R. Dynamic network embedding survey. *Neurocomput. 472*, C (feb 2022), 212–223.

[75] YAO, L., MAO, C., AND LUO, Y. Graph convolutional networks for text classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence* (2019), AAAI'19/IAAI'19/EAAI'19, AAAI Press.

[76] ZHANG, X., ZHANG, L., JIN, B., AND LU, X. A multi-view confidence-calibrated framework for fair and stable graph representation learning. In *2021 IEEE International Conference on Data Mining (ICDM)* (2021), pp. 1493–1498.

[77] ZHENG, X., WU, B., ZHANG, A. X., AND LI, W. Improving robustness of gnn-based anomaly detection by graph adversarial training. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)* (2024), pp. 8902–8912.

[78] ZHOU, J., CUI, G., HU, S., ZHANG, Z., YANG, C., LIU, Z., WANG, L., LI, C., AND SUN, M. Graph neural networks: A review of methods and applications. *AI open 1* (2020), 57–81.

[79] ZHOU, J., CUI, G., ZHANG, Z., YANG, C., LIU, Z., AND SUN, M. Graph neural networks: A review of methods and applications. *ArXiv abs/1812.08434* (2018).

[80] ZHU, L., GUO, D., YIN, J., STEEG, G. V., AND GALSTYAN, A. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering 28*, 10 (2016), 2765–2777.