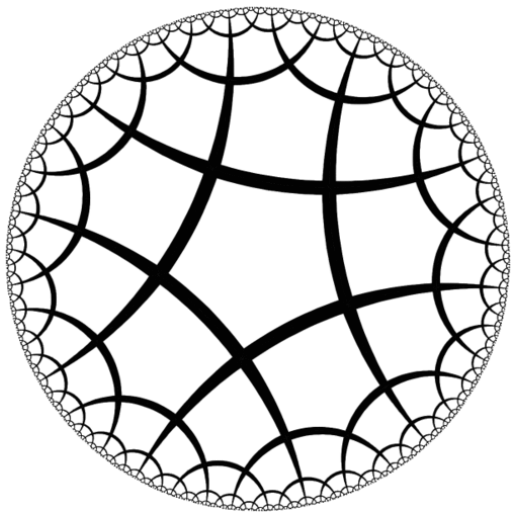

**Software Engineering Department**

**Capstone Project Phase A**

# Citation Anomaly Detection via Lorentzian Embeddings and Temporal Attention

**25-2-R-3**



**Nadir Yaakov**
**Ben Zacai**

**Supervisor: Dr. Dvora Toledano-Kitai**
**GitHub**

# Table of Contents

## Abstract

Citation networks provide valuable insight into the evolution of knowledge through scholarly publications, where nodes represent academic papers and edges denote citation relationships. Detecting anomalies within such networks, such as sudden citation spikes, excessive self-citation, or unusually dense citation clusters, can indicate artificial influence, scholarly misconduct, or emerging research trends. However, traditional graph analysis techniques and shallow embedding methods often fail to capture the complex, hierarchical, and temporal nature of citation data.

To address this limitation, we propose a targeted anomaly detection framework that extends the Lorentzian Linear Graph Convolutional Network ($L^2GC$) to dynamic citation networks. $L^2GC$ is a graph neural network architecture operating in hyperbolic space, making it well-suited for modeling the tree-like structures commonly observed in citation graphs. Our approach segments the evolving citation network into a sequence of time-based snapshots and applies $L^2GC$ independently to each in order to capture structural and temporal deviations over time.

To support unsupervised anomaly detection, the hyperbolic node embeddings are projected back into Euclidean space using logarithmic maps and subsequently analyzed using the Isolation Forest algorithm. For validation, we introduce synthetic anomalies through controlled noise injection and evaluate the model's performance using standard metrics, including precision, recall, and F1 score.

This framework enables the detection of subtle, time-dependent anomalies in evolving citation networks, offering a scalable and geometry-aware tool for monitoring research integrity.

## 1. Introduction

The analysis of scholarly citation networks has become a crucial methodology in understanding how knowledge is produced, disseminated, and evaluated within the academic community. By representing papers as nodes and citations as directed edges, these networks capture the intricate dynamics of scholarly communication, enabling the identification of patterns related to academic influence, intellectual evolution, and research trends (Radicchi, Fortunato, & Vespignani, 2012).

However, as citation networks continue to expand in both size and complexity, new challenges emerge, particularly in identifying atypical or anomalous citation behavior. Such anomalies may include sudden citation spikes, excessive self-citation, or the emergence of artificially dense citation clusters. These patterns could signal scientific misconduct, attempts to manipulate bibliometric indicators, or conversely, may reflect emerging trends or overlooked but influential research contributions (Ciotti et al., 2016).

Traditional graph analysis techniques and shallow embedding methods, such as matrix factorization or random walk-based models, often fall short in capturing the rich hierarchical and temporal properties inherent in real-world citation networks (Goyal & Ferrara, 2018; Xu, 2021). Recent advancements in Graph Neural Networks (GNNs) and hyperbolic representation learning offer more expressive and structured frameworks for modeling complex network data (Zhang et al., 2019). Hyperbolic geometry aligns well with the tree-like and hierarchical nature of citation graphs, enabling better preservation of global relationships in lower-dimensional space (Cannon et al., 1997).

One notable model in this domain is the Lorentzian Linear Graph Convolutional Network ($L^2GC$), which operates in Lorentzian hyperbolic space and has demonstrated strong performance in representing static graphs with hierarchical structure (Liang, Wang, Bao, & Gao, 2024). However, its current formulation is limited to static graphs and does not account for the evolving nature of citation networks over time.

This project proposes a focused framework for anomaly detection in evolving citation networks by extending the $L^2GC$ model, originally designed for static graphs, into a temporal framework. Specifically, the approach suggests segmenting the full citation network into a sequence of time-based snapshots (e.g., by year), applying the $L^2GC$ model independently to each temporal subgraph, and tracking the evolution of node embeddings over time. This temporal decomposition enables the detection of anomalies that emerge not only from unusual network structure but also from abrupt shifts in citation dynamics (Xu, Ruan, Korpeoglu, Kumar, & Achan, 2020).

To further enhance interpretability and enable precise anomaly scoring, the learned hyperbolic embeddings are transformed back to Euclidean space using logarithmic maps, followed by applying the Isolation Forest algorithm to detect outlier behavior.

To validate the proposed method, we introduce a controlled noise injection strategy, simulating anomalous behavior by adding synthetic citation patterns to the full network. Performance is measured using standard metrics such as precision, recall, and F1, providing a quantitative basis for validating anomaly detection under realistic and noisy conditions with further analysis of false positives and temporal consistency. By integrating temporal modeling, hyperbolic geometry, and unsupervised learning, this project offers a robust, scalable, and hyperbolic-geometry-aware framework for identifying subtle, time-dependent anomalies in citation networks, enhancing the understanding of scholarly behavior and contributing to research integrity at scale.

The remainder of this document is organized as follows: Section 2 provides theoretical background on graph theory, citation networks, graph embedding techniques, and graph neural networks, including an introduction to hyperbolic geometry and anomaly detection. Section 3 introduces the mathematical foundations of the Lorentz model of hyperbolic geometry and the $L^2GC$ model. Section 4 describes the proposed approach for extending $L^2GC$ to dynamic citation networks through temporal segmentation and anomaly detection. Section 5 outlines the validation methodology, including synthetic noise injection and

expected performance metrics. Section 6 details the tools and technologies required for implementation. Finally, Section 7 presents a reflection on challenges encountered in Phase A and considerations for future implementation.

## 2. Theoretical Background

### 2.1 Graph Theory and Networks

A graph is a mathematical structure formally defined as a set of nodes (vertices) and edges that connect pairs of nodes. In simpler terms, it provides a way to represent relationships between entities (Van Steen, 2010).

A network is an informal term for a large, interconnected collection of nodes, where each node represents an entity, such as people, computers, or organizations, and the edges represent relationships or interactions between them. Due to their scale and complexity, analyzing networks requires methods that go beyond examining individual components, instead focusing on global structural properties.

While graphs and networks provide a foundational framework for representing relationships between entities, they can exhibit significantly different behaviors over time. Broadly, networks can be classified as static or dynamic, depending on whether their structure remains unchanged or evolves over time. Static networks maintain a fixed topology, where nodes and edges do not change once the network is constructed. In contrast, dynamic networks continuously evolve, with nodes and edges being added, removed, or modified over time.

Feature vectors provide a numerical representation of nodes within a network by encoding their structural and/or attribute-based properties, encapsulating various attributes of them to facilitate network analysis. These vectors serve as essential inputs for a wide range of machine learning tasks, including link prediction, node classification, and anomaly detection. Depending on the nature of the network, feature vectors may incorporate various types of information, such as topological features (e.g., degree, centrality measures, clustering coefficients), node metadata, or content-based features derived from textual data.

#### 2.1.1 Citation Networks

Citation networks represent academic papers and their citation relationships as directed graphs, where nodes correspond to scientific papers or authors and edges indicate citations. A directed edge points from the citing paper to the cited one, reflecting the flow of knowledge and influence (Radicchi, Fortunato, & Vespignani, 2012).

These networks are typically constructed from academic datasets containing bibliographic information, including citation records. A key example is a paper citation network, where each directed edge represents a citation from one paper to another, forming a structured web

of scholarly references. Analyzing these networks using graph-based models provides insights into academic impact, knowledge diffusion, and research trends.

In the context of citation networks, feature vectors typically integrate both structural and textual features of academic papers. Textual features, such as titles, abstracts, and keywords, can be extracted using methods like TF-IDF, word embeddings, or language models. Structural features might include in-degree (the number of citations a paper has received), out-degree (the number of citations a paper has), metrics like betweenness and centrality, or local clustering coefficients. By combining these features, machine learning models can effectively predict future potential citations and uncover meaningful patterns in scholarly communication. For instance, similarity measures like the Jaccard coefficient and cosine similarity have been shown to be influential in predicting citation links (Shibata et al., 2011), and the tendency of similar papers to cite one another reflects homophily in knowledge diffusion (Ciotti et al., 2016).

### 2.2 Graph Embedding

Graph embedding is a technique used to represent graph-structured data in a low-dimensional vector space while preserving its structural and relational properties (Goyal & Ferrara, 2018). This transformation enables the application of traditional machine learning algorithms for tasks like link prediction and anomaly detection. By transforming graph data into a continuous and dense vector space, embedding enables deeper understanding of network properties, enhances interpretability, and improves the performance of downstream analytical tasks (Xu, 2021).

The task of obtaining a meaningful vector representation of graph nodes is challenging, presenting several key considerations for researchers. These challenges include:

- Preservation of Graph Properties: A fundamental challenge lies in determining which structural and relational properties of the original graph should be preserved in the lower-dimensional embedding space. The choice of property can significantly impact the utility of the embeddings for downstream tasks and may include first-order proximity (direct links), second-order proximity (shared neighbours), or higher-order structural similarities.
- Scalability to Large Networks: Real-world networks often comprise millions of nodes and edges, requiring embedding techniques that can scale efficiently to handle such massive datasets without excessive computational costs.
- Selection of Embedding Dimensionality: Determining the optimal dimensionality of the embedding space is a non-trivial problem. Insufficient dimensions may lead to information loss, while excessive dimensions can increase computational complexity and potentially introduce noise. The ideal dimensionality can also be application-specific.
- Handling Diverse Graph Characteristics: Graphs in real-world scenarios exhibit a wide range of characteristics, including directed and undirected edges, weighted

connections, and diverse node and edge types. Embedding techniques need to be adaptable to these diverse structures.

Graph embedding techniques have emerged as an important approach in analyzing complex network data, which is widespread in various real-world domains such as social networks, biological systems, and communication networks.

Graph embedding techniques are applied to both static and dynamic networks, which differ in their structural properties and temporal evolution.

To overcome the challenges associated with graph representation, a wide range of embedding techniques has been developed, categorized based on their underlying methodologies.

### 2.2.1 Embedding in Static Networks

As mentioned, static networks maintain a fixed structure over time, with edges and nodes remaining unchanged. These networks are analyzed using traditional embedding techniques, which fall into several categories:

- **Factorisation-Based Methods:** These techniques leverage matrix decomposition to generate embeddings. An example is HOPE (High-Order Proximity Preserved Embedding) (Goyal & Ferrara, 2018), which preserves higher-order proximities by applying generalized Singular Value Decomposition (SVD).
- **Random Walk-Based Methods:** These methods generate sequences of nodes using random walks and learn embeddings based on node co-occurrence. Notable examples include DeepWalk, node2vec (Goyal & Ferrara, 2018), and LINE (Xu, 2021) , which balance breadth-first and depth-first search strategies to capture network structures effectively.
- **Deep Learning-Based Methods:** These methods apply neural network architectures for representation learning. Examples include Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), which model complex structural dependencies in the graph.

### 2.2.2 Embedding in Dynamic Networks

Dynamic networks evolve over time, with nodes and edges being continuously added, removed, or modified. These networks are prevalent in social network interactions, where friendships and interactions change continuously, financial transaction networks that exhibit temporal fluctuations, and academic citation networks, where papers are continuously published, adding new citations and modifying the connectivity of the network over time. Capturing temporal dependencies requires specialized embedding, including:

- **Snapshot-Based Methods:** These techniques construct embeddings by capturing the network state. A notable example is DynamicTriad, which models the evolution of triadic relationships over time.

- **Deep Learning and Self-Attention-Based Methods:** These approaches leverage deep learning and attention mechanisms to dynamically update node embeddings. For example, EvolveGCN dynamically adapts the parameters of a graph convolutional network (GCN) in a time-dependent manner.

By integrating these embedding techniques, we can gain deeper insights into network structures and temporal patterns, improving the accuracy of anomaly detection, link prediction, and network evolution modeling.

A more recent paradigm in graph embedding involves representing nodes as Gaussian distributions in a latent space rather than as single-point vectors. This approach enables the modeling of uncertainty in embeddings and captures richer semantic relationships within the graph structure.

- Graph2Gauss (G2G) (Goyal & Ferrara, 2018) is an unsupervised and inductive model that learns Gaussian embeddings for attributed and directed/undirected graphs by employing multi-hop neighbourhood sampling and a deep encoder.
- Deep Variational Network Embedding (DVNE) (Goyal & Ferrara, 2018) uses deep variational autoencoders to generate Gaussian embeddings, optimizing a hybrid loss function that preserves both first-order and second-order proximities, thereby enhancing the representation of complex network structures.

Graph embeddings play a crucial role in anomaly detection within dynamic citation networks by capturing both structural and temporal patterns. By learning representations that preserve the evolving nature of citation relationships, embeddings enable the identification of unusual citation behaviors, such as sudden citation spikes, isolated clusters, or irregular citation patterns. This approach enhances the ability to detect influential breakthroughs, citation manipulation, and overlooked yet significant research, contributing to a deeper understanding of scholarly network dynamics.

### 2.3 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are deep learning models designed to extract meaningful representations from graph-structured data, capturing complex relationships that traditional methods often overlook (Zhang et al., 2019). They are widely used in domains such as social analysis, bioinformatics, and computer vision, where structural dependencies play a crucial role. By representing data as graphs, GNNs can encode this structural information to gain deeper insights compared to analyzing isolated data points. Unlike shallow graph embedding techniques, GNNs leverage deep learning to learn low-dimensional node or graph representations while preserving essential graph properties.

A prominent type of GNN is the Graph Convolutional Networks (GCN), which aggregates information from a node's neighbourhoods in a convolutional manner. GCNs have shown great expressive power in learning graph representations and have achieved state-of-the-art results in a wide range of tasks and applications.

Existing Graph Convolutional Network (GCN) models can be broadly classified into two main categories based on their convolution approach:

- Spectral-based GCNs: These methods define graph convolutions from a spectral perspective, drawing an analogy to classic signal processing using the graph Fourier transform. Spectral graph convolutions are computed by taking the inverse Fourier transform of the multiplication between the Fourier transformed graph signals and a filter in the spectral domain. Models by (Bruna et al., 2014) and (Henaff et al., 2015) laid the groundwork for spectral-based convolutions. ChebNet, introduced by (Defferrard,2017), improves localization by using K-polynomial filters and employing Chebyshev polynomial approximation, reducing computational complexity. A widely used variant, the GCN model by (Kipf et al., 2017), simplifies ChebNet by approximating the Chebyshev polynomial to the first order, leading to a convolution operation that effectively aggregates representations from direct neighbours. This GCN model is often seen as bridging the gap between spectral and spatial methods.
- Spatial-based GCNs: These methods perform convolution directly in the spatial domain, aggregating node features from their nodes' neighborhoods. As described in (Zhang et al., 2019), approaches such as PATCHY-SAN apply convolutional neural network-like operations to graphs by extracting and ordering local neighbourhoods, LGCN transforms graph data into a grid-like structure, and GraphSAGE introduces learnable aggregation functions for inductive learning. Other notable models include DCNN, which uses graph diffusion for feature propagation, and MoNet, which defines a general framework based on pseudo-coordinates.

GCNs are widely used in computer vision, NLP, and social network analysis for tasks like classification, link prediction, and recommendation. However, challenges remain in developing deeper architectures, handling dynamic graphs, and enhancing expressive power, driving ongoing research in graph-based deep learning.

### 2.4 Hyperbolic Geometry

Hyperbolic geometry is a form of non-Euclidean geometry that emerged from questioning Euclid's parallel postulate (CANNON et al., 1997). In contrast to Euclidean geometry, where exactly one parallel line can be drawn through a point external to a given line, hyperbolic geometry allows infinitely many such parallel lines (CANNON et al., 1997). This deviation introduces a fundamentally different geometric structure characterized by constant negative curvature (Peng et al., 2021; CANNON et al., 1997).

The negative curvature of hyperbolic space gives rise to several properties that sharply contrast with those of flat Euclidean space (Peng et al., 2021; CANNON et al., 1997). For instance, in Hyperbolic space:

- The circumference and area of a circle grow exponentially with the radius, rather than linearly or quadratically as in Euclidean geometry.

- The sum of the angles in a triangle is always less than π (180 degrees), and the area of a triangle is proportional to its angular deficit.
- Similar triangles are congruent, meaning they have the same size and shape, unlike in Euclidean geometry, where similar triangles may differ in scale.

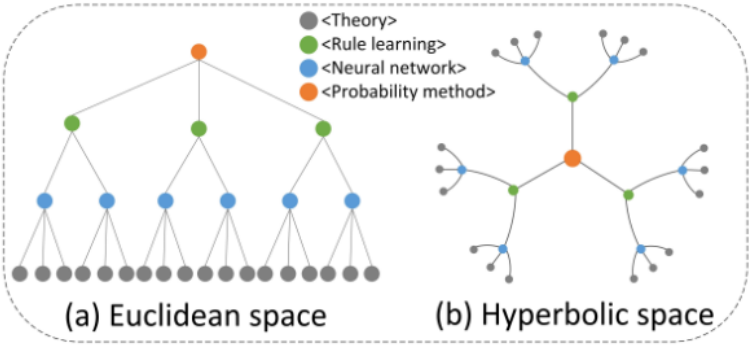



Figure 1: Visualization of a tree-like hierarchical structure citation network in Euclidean and hyperbolic space. In both graphs, nodes represent documents, and edges represent citations. Each color node represents a type of document (Liang, Wang, Bao, & Gao, 2024).

Several models exist to represent hyperbolic space (Peng et al., 2021; CANNON et al., 1997), including the Hyperboloid Lorentz model, the Poincaré ball model, the Poincaré half-space model, and the Klein model. The term *hyperbolic geometry* is derived from the hyperboloid model, which uses hyperbolas to define distance and curvature in space.

In recent years, hyperbolic geometry has found increasing relevance in machine learning, particularly for representing data with hierarchical or tree-like structures (Peng et al., 2021). The exponential growth of volume in hyperbolic space aligns naturally with hierarchical data, making it ideal for embedding tasks in fields such as natural language processing, graph representation, and single-cell transcriptomics.

This development led to the emergence of Hyperbolic Deep Neural Networks (HDNNs), which leverage the unique geometric properties of hyperbolic space for tasks like natural language processing, graph analysis, and single-cell analysis. Unlike traditional Euclidean-based models that use inner products to measure similarity, models built in hyperbolic geometry incorporate curvature into distance calculations. This helps models with capturing hierarchical and relational structure more effectively, performing dimensionality reduction while preserving global structure, and accelerating training and improving predictive performance.

A foundational aspect of these approaches is the use of linear and nonlinear transformations that embed high-dimensional data into hyperbolic space while preserving geometric relationships, particularly in graph-based representations (Peng et al., 2021).

### 2.5 Anomaly detection

Graph-based anomaly detection techniques analyse both the structural and attribute-based characteristics of graphs to identify irregular or deviant patterns. These methods span a spectrum from basic analyses of node degree distributions and connectivity patterns to more advanced approaches that leverage graph embedding and machine learning models.

In the specific context of academic citation networks, anomaly detection aims to uncover atypical behaviors associated with papers, author nodes, or citation links that deviate from normative citation patterns. Notable examples include papers citing an unusually high number of unrelated sources, artificially dense citation clusters among selected groups of authors, or excessive self-citation patterns that may raise concerns. Such anomalies may indicate attempts to manipulate bibliometric indicators, such as the h-index, or can reflect non-authentic academic activity (Yan et al., 2012; Li et al., 2021). Early detection of these irregularities is essential for ensuring the integrity, authenticity, and scientific value of scholarly publications.

## 3. Preliminaries

### 3.1 The Lorentz Model of Hyperbolic Geometry

The Lorentz model is one of the standard representations of n-dimensional hyperbolic space, characterized by constant negative curvature $k < 0$. It is formally defined as a Riemannian manifold $L_k^n = (L^n, g_k^x)$, where $L^n \subset \mathbb{R}^{n+1}$ is the upper sheet of a two-sheeted hyperboloid in Minkowski space and $g_k^x$ denotes the Riemannian metric tensor given in this model by $g_k^x = \mathrm{diag}(-1, 1, \ldots, 1)$.

The point set of the Lorentz model is defined as:

$$L^n = \{x \in \mathbb{R}^{n+1} \mid \langle x, x \rangle_L = 1/k, x_0 > 0\}$$

where $\langle x, y \rangle_L$ is the Lorentzian Inner Product, defined

for any two points $x = (x_0, \ldots, x_n)$ and $y = (y_0, \ldots, y_n)$ in the Lorentz space, as:

$$\langle x, y \rangle_L = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i$$

This inner product induces the geometry of hyperbolic space within the ambient $(n+1)$-dimensional Minkowski space. The geodesic distance between two points in the Lorentz model reflects the intrinsic hyperbolic geometry and typically exceeds the corresponding Euclidean distance after embedding.

The hyperbolic distance between them is then calculated by:

$$\underline{d_L{}^k(x,y) = \sqrt{\frac{1}{-k}} cosh^{-1}(k \cdot \langle x,y\rangle_L)}\,,$$

where $k < 0$ is the negative constant curvature of the space. This distance preserves the hierarchical and tree-like relationships inherent in citation graphs more effectively than Euclidean distance.

To facilitate computations involving hyperbolic embeddings, the Lorentz model employs exponential and logarithmic maps to move between the hyperbolic manifold and the tangent space at a given point.

The Lorentz model utilizes exponential and logarithmic maps for transformations between the hyperbolic space and the tangent space.

- **Exponential Map**: Given a tangent vector $z \in T_x L_k{}^n$, the exponential map at point $x$ maps it to the hyperbolic space $L_k{}^n$ along a geodesic and is defined as:

  $$\exp_x^k(z) = \cosh(\alpha)x + \sinh(\alpha)\frac{z}{\alpha},$$

  where $\alpha = \sqrt{-k\|z\|_L}$ and $\|z\|_L = \sqrt{\langle z,z\rangle_L}$.

- **Logarithmic Map:** The inverse mapping from a point $y \in L_k{}^n$ in the hyperbolic space back to the tangent space $T_x L_k{}^n$ at x is given by:

  $$\log_k{}^x(y) = \frac{cosh^{-1}(\beta)}{\sqrt{\beta^2 - 1}} \cdot (y - \beta x),$$

  where $\beta = k\langle x, y\rangle_L$.

These maps are fundamental tools for differential operations and learning algorithms defined in hyperbolic space, allowing for efficient modeling of hierarchical and tree-like data structures.

- **Canonical Function Transfer Between Lorentzian Hyperbolic Spaces:** Transferring functions between two hyperbolic spaces, $L_k^n$ and $L_k^m$ where $k < 0$ denotes the constant negative curvature, can be achieved using exponential and logarithmic maps centered at the origin. Given a function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$, its canonical Lorentzian extension to the hyperbolic setting, denoted as $f^{\otimes^k}$, is defined as:

  $$f^{\otimes^k}(\mathrm{x}) = exp_0^k(\,\hat{f}\,(\,log_0^k(\mathrm{x})\,)\,)$$

Here, $\mathbf{x} \in L_k^n$ is a point in the source hyperbolic space, and $log_0^k(\mathbf{x}) \in T_0 L_k^n$ maps this point to the tangent space at the origin. The intermediate function $\hat{f}$ applies the Euclidean transformation $f$ only to the spatial coordinates $(v_1, ..., v_n)$ while setting the temporal component $v_0 = 0$. That is:

$$\hat{f}(v) = (0, f(v_1, ..., v_n))$$

Finally, the exponential map $exp_0^k$ maps the transformed point back to the target hyperbolic space $L_k^m$.

This formulation preserves the geometric structure of the hyperbolic spaces while enabling the application of Euclidean transformations in a curved setting. It is particularly useful in neural architectures and embedding techniques operating in the Lorentzian model of hyperbolic space.

The Lorentz model plays a central role in the architecture of Lorentzian Linear Graph Convolutional Networks (L²GC ), which extend conventional graph convolutional networks (GCNs) to hyperbolic space. In this setting, the Lorentz model is used to better capture the tree-like hierarchical structure of complex networks, offering improved representational capacity over Euclidean alternatives.

### 3.2 The L²GC Model

The Lorentzian Linear Graph Convolutional Network (L²GC ) is designed to generalize linear graph convolution operations to hyperbolic space, leveraging the representational power of the Lorentz model to encode hierarchical and tree-like structures more effectively. The L²GC framework operates in three main stages, combining both Euclidean and hyperbolic computations:

#### 3.2.1 Parameter-Free Neighborhood Feature Propagation in Euclidean Space

This step constitutes the first step of the L²GC framework. Its goal is to balance the influence of the graph structure and the original node features without relying on any trainable parameters. The procedure is as follows:

- A self-looped graph $\tilde{G}$ is created by adding self-connections to the original graph $G = (V, E)$ with $n$ nodes and $m$ edges.

- A linear propagation matrix $P$ is computed as

  $$P = (D + I)^{-1/2} (A + I) (D + I)^{-1/2},$$

  where $A$ is the adjacency matrix $G$ and $D$ is the diagonal degree matrix $G$.

- Node features are updated using the information transfer between the node and its $n$-power neighbors :

  $$H^{(l+1)} = (1 - \alpha) P H^{(l)} + \alpha X,$$

where $X \in \mathbb{R}^{n \times d}$ is the initial node feature matrix, where each node $v \in V$ is associated with a feature vector $x_v \in \mathbb{R}^d$, and $\alpha \in [0,1]$ is a teleport (or retention) probability that controls the influence of the original features at each propagation step, and $l$ denotes the layer or iteration index.

- This iterative process ensures that each layer representation $H^{(l)}$ is influenced by both the graph structure and the initial node features $X$, with a fixed proportion. As a result, the final node representations $H^{(n)}$ are derived through $n$-step propagation, capturing both local and global structural information. This parameter-free propagation scheme blends the initial node features with information from the graph structure at each step, thereby mitigating the over-smoothing effect commonly observed in deep graph convolutional networks.

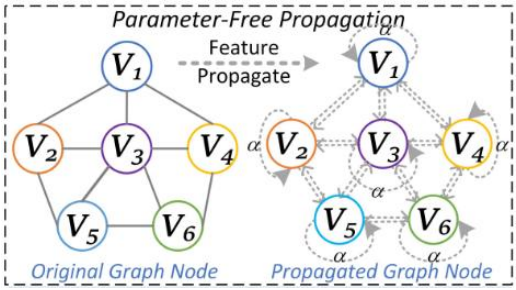



Figure 2: The Parameter-Free Propagation Process (Liang, Wang, Bao, & Gao, 2024).

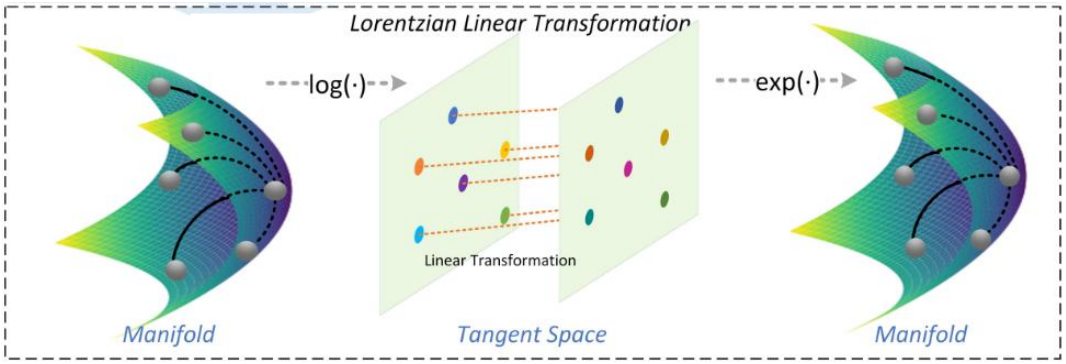



Figure 3: The framework of Lorentzian Linear Graph Convolutional Networks for node Classification (Liang, Wang, Bao, & Gao, 2024).

### 3.2.2 Lorentzian Linear Feature Transformation in Hyperbolic Space

After the feature propagation stage, the learned Euclidean node representations $H^{(n)}$ are mapped into Lorentzian hyperbolic space using the exponential map at the origin, denoted $exp_k^0$. This operation yields the initial hyperbolic embeddings:

$$x^{(n)} = exp_k^0(H^{(n)})$$

Once embedded in hyperbolic space, the L²GC model performs a Lorentzian linear transformation, which serves as the hyperbolic analogue of a standard linear layer in Euclidean GCNs. This transformation is performed through the following sub-steps:

- Logarithmic map: The hyperbolic embeddings $x^{(n)}$ are first mapped back to the tangent space at the origin:

$$v = log_k^0(x^{(n)})$$

- Linear transformation in tangent space: A Euclidean linear transformation is then applied to the spatial (space-like) components ($v_1, \ldots, v_n$) of the tangent vectors, while leaving the time-like coordinate unchanged:

$$\widehat{M}(v) = (0, M(v_1, \ldots, v_n))$$

  Here $M \in \mathbb{R}^{m \times n}$ is a learnable weight matrix representing a linear transformation $M$: $\mathbb{R}^n \rightarrow \mathbb{R}^m$, applied only to the spatial part in accordance with Lorentzian geometry.

- Exponential map: The transformed vectors are then mapped back to the hyperbolic Lorentz manifold:

$$M^{\otimes}(v) = \exp_k^0(\widehat{M}(v))$$

To summarize the full transformation pipeline, from Euclidean features to their hyperbolic representations after linear transformation, the operations are composed as follows

$$M^{\otimes}(H^{(n)}) = \exp_k^0(\widehat{M}(\log_k^0(exp_k^0(H^{(n)}))))$$

This composite formulation encapsulates the entire process while preserving the geometric consistency of the Lorentzian manifold, thereby enabling expressive and curvature-aware feature transformations within the hyperbolic space.

### 3.2.3 Node labels prediction in Euclidean space

In its original formulation, the L²GC framework concludes by mapping the transformed hyperbolic embeddings d back to Euclidean space using the logarithmic map

$$z = log_k^0(x_{transformed})$$

This mapping allows the application of standard classifiers (e.g., SoftMax layers) for node label prediction. Returning to Euclidean space, ensures compatibility with common loss functions and evaluation metrics, facilitating supervised learning tasks..

By incorporating a Lorentzian linear transformation in hyperbolic space, L²GC leverages the expressive power of non-Euclidean geometry, particularly its capacity to model hierarchical and tree-like structures with lower distortion. As a result, L2GC demonstrates strong performance on datasets characterized by low Gromov $\delta$-hyperbolicity, such as citation networks.

## 4. The Approach

In this project, we propose an alternative anomaly detection framework for academic citation networks by adapting the Lorentzian Linear Graph Convolutional (L²GC) model to a temporal setting. While L²GC was originally designed to operate on static graphs and to perform node classification by mapping hyperbolic embeddings back to Euclidean space for supervised learning, our approach diverges from both aspects.

First, we extend the model to operate on dynamic citation networks by segmenting the graph into a sequence of temporal subgraphs (e.g., per year) and applying L²GC independently to each snapshot: This temporal decomposition allows the model to capture time-dependent structural changes and detect time-dependent structural anomalies/capabilities that are not available in the original static formulation.

Second, instead of pursuing node classification, we utilize the expressive hyperbolic embeddings generated by L²GC as input for unsupervised anomaly detection. By tracking the temporal evolution of node embeddings across snapshots and applying methods such as Isolation Forest, we identify papers exhibiting unusual citation behavior over time. This shift in both temporal modeling and analytical objective allows us to preserve the geometric advantages of L²GC while tailoring the framework to the unique challenges of anomaly detection in dynamic citation networks.

### 4.1 Input: Citation Network

We model the academic citation network $G = (V, E)$ as a *temporal graph*, where $V$ represents academic papers (nodes) and $E$ denotes time stamped citation links (edges). Each citation event carries a time label (e.g., year of publication), enabling a dynamic view of the network rather than a static aggregate. The graph evolves over time, with citation relationships changing as new papers are published and cite existing work. The goal is to detect anomalous citation patterns, which may indicate unusual citation behavior or suspicious behavior in the scholarly record.

### 4.2 Data Preparation and Temporal Segmentation

#### 4.2.1 Data Preparation

Before applying the model to dynamic citation networks, we construct a static feature representation for each node based on its intrinsic attributes. Each node, corresponding to a

scholarly paper, is assigned a fixed feature vector that encodes time-invariant information such as publication metadata, citation count, venue, and authorship. These vectors serve as the foundational input for subsequent embedding stages and remain unchanged across all temporal segments.

### 4.2.2 Temporal Segmentation

To model the dynamic nature of the citation network, we divide the citation timeline into a sequence of discrete intervals (e.g., yearly or multi-year windows). For each time window $t$ we construct a corresponding temporal subgraph $G_t = (V_t, E_t)$, where $V_t$ includes all papers published up to and including time $t$, and $E_t$ includes all citations that occurred up to and including the same time. This cumulative construction ensures that each subgraph reflects the complete historical state of the network at that moment.,

This segmentation preserves the chronological order of citation events and provides a time-resolved view of the network's development. By comparing structural patterns across successive time windows, the model can identify temporal anomalies, such as abrupt shifts in connectivity or unusual citation bursts, that would remain hidden in a static, aggregated representation. As such, this approach explicitly captures the dynamic nature of citation behavior and enhances the model's ability to detect evolving structural irregularities over time.

Building on this temporally segmented structure, we next apply the L²GC model independently to each snapshot, enabling the extraction of time-aware node representations that reflect both local and global structural changes over time.

## 4.3 L²GC Model on Temporal Subgraphs

To adapt the L²GC model to a dynamic citation network, we apply it separately to each temporal subgraph $G_t = (V_t, E_t)$ . The goal is to generate hyperbolic node embeddings that capture the local and global structure of the network within each time window. This process proceeds through the following key sub-steps:

**Step 1: Static feature initialization**

Each node $v \in V_t$ is associated with a static feature vector $f_v \in \mathbb{R}^k$, representing time-invariant attributes such as publication metadata, citation count, or venue information. These features, resulting in a feature matrix $X$, are computed once during preprocessing and shared across all subgraphs to ensure consistency.

**Step 2: Projection into hyperbolic space**

To enable learning in a geometry that better captures the hierarchical and scale-free structure of citation networks, the static feature vectors obtained from step 1 are projected into the Lorentzian hyperbolic space via the following two sub-steps:

- Each feature vector is interpreted as a vector in the tangent space.

The feature vectors are projected into the Lorentzian hyperbolic space through a two-step process:

- First, the vectors are projected to the tangent space at the origin of the Lorentzian manifold. This step enables compatibility between the Euclidean embedding and the geometry of the hyperbolic space by ensuring that computations begin in a flat, local approximation of the manifold.
- The tangent vector is then mapped onto the Lorentzian hyperbolic manifold using the exponential map. This transformation ensures that the resulting point lies on the manifold itself and preserves the geometric structure necessary for downstream operations in L²GC.

  This two-step process ensures that the learned representations fully reside in the Lorentzian space and are ready to be processed by the subsequent convolutional layers, which are designed to operate in non-Euclidean geometries.

**Step 3: Lorentzian graph convolution**

Within each temporal subgraph, L²GC performs a graph convolution tailored to hyperbolic space. The model aggregates information from a node's local neighborhood using Lorentzian linear transformations, which preserve the manifold's geometry and efficiently capture hierarchical patterns.

**Step 4: Independent application per time window**

The entire process is executed independently for each subgraph $G_t$, producing a unique set of node embeddings $X_t$ per time interval. This enables the model to capture structural evolution over time and forms the basis for temporal anomaly detection.

This five-step process extends the L²GC model from static to dynamic settings while preserving its geometric strengths, allowing for meaningful analysis of evolving citation behavior across time.

### 4.4 Tracking Temporal Feature Evolution

Once node embeddings have been generated for each temporal subgraph via L²GC, we obtain a sequence of feature representations for every node over time. These embedding trajectories reflect how each node's structural role within the citation network evolves across consecutive time windows. Analyzing these trajectories enables the detection of abnormal changes that may signify anomalous citation behavior. Under typical conditions, a node's embedding changes gradually as the network evolves. However, a sudden shift, such as a spike in citations, can result in a significant displacement in embedding space, indicating potential irregularity. This approach aligns with techniques in temporal graph neural networks, which monitor the evolution of node-level features across snapshots.

The process is composed of the following key sub-steps:

**Step 1: Constructing Sequential embeddings:**

For each time window $t \in (1,2,\ldots,T)$, let $X_t$ denote the matrix of node embeddings produced by L²GC for the subgraph $G_t$. For each node $v$, this forms a temporal embedding sequence, $x_v = (x_{v,1}, x_{v,2}, \ldots, x_{v,T})$ representing its position in hyperbolic space over time.

**Step 2: Measuring change between time steps:**

To identify unexpected shifts in a node's behavior, we compare its embeddings at successive time steps, e.g., compute the Euclidean distance between $x_{v,t}$ and $x_{v,t-1}$. Large deviations may indicate citation anomalies, such as sudden increases in influence or unusual citation patterns.

**Step 3: Analyzing dynamic patterns**

Examining the shape and variability of each node's embedding trajectory allows us to detect evolving patterns and concept drift in the network. For example, a paper that suddenly garners many citations after being obscure will have an abrupt embedding change in the corresponding time window.

Tracking node embeddings over time reveals when and where structural roles in the citation graph diverge from expected behavior. These temporal deviations provide a rich signal for subsequent anomaly scoring, as detailed in the next section

### 4.5 Anomaly Scoring with L²GC

To quantify deviations in citation behavior, we compute anomaly scores based on the node embeddings produced by L²GC for each temporal subgraph. Since L²GC operates in Lorentzian hyperbolic space, we first map the embeddings back to Euclidean space using the logarithmic map at the origin. This enables the use of standard anomaly detection algorithms in a familiar geometric setting.

We adopt the Isolation Forest (IForest) algorithm due to its effectiveness in identifying outliers in high-dimensional spaces without supervision. The anomaly scoring process proceeds through the following sub-steps:

**Step 1: Back-projection to Euclidean space**

For each temporal subgraph $G_t$, we map the L²GC node embeddings $x_{v,t}$ from Lorentzian space to Euclidean space using the logarithmic map:

$$z_{v,t} = log_k^0(x_{v,t})$$

This transformation preserves relative distances while enabling compatibility with Euclidean-based algorithms.

**Step 2: Local anomaly detection using IForest**

We apply the Isolation Forest algorithm separately to the embeddings $\{z_{v,t}\}_{v \in V_t}$ in each time window. For every node $v$, IForest assigns an anomaly score $a_{v,t} \in \mathbb{R}$, indicating how isolated or atypical the node is compared to its peers within that subgraph.

**Step 3: Constructing temporal anomaly profiles**

By collecting anomaly scores across all time windows, we obtain a temporal anomaly vector for each node:

$$a_v = (a_{v,1}, a_{v,2} \ldots, a_{v,T})$$

This vector captures the evolution of the node's anomaly level over time and serves as the basis for detecting both persistent and sudden anomalies.

**Step 4: Anomaly interpretation and detection strategies**

We suggest analyzing the temporal anomaly vectors using the following strategies:

- Aggregating scores (e.g., max or average) to identify consistently anomalous nodes;
- Detecting sudden spikes or unusual patterns in time;
- Comparing anomaly trends across nodes to identify group behaviors or structural shifts.

By combining L²GC's representational power with temporal anomaly scoring, this approach enables the detection of subtle, evolving, and context-dependent anomalies in dynamic citation networks, such as suspicious citation bursts, emerging manipulation patterns, or anomalous research impact.

# Full process overview using L2GC

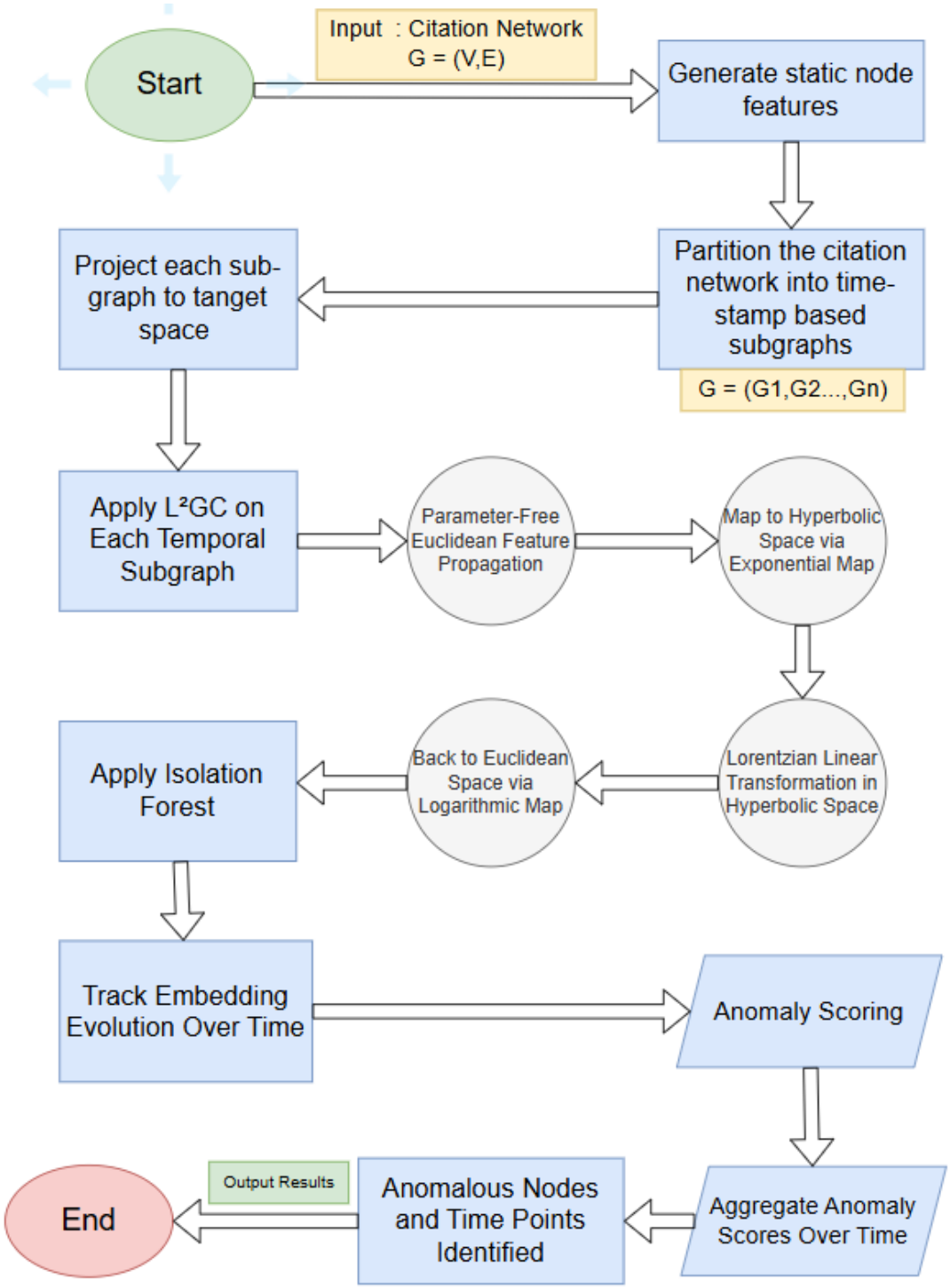



Figure 4: Full process of finding anomalies from the given citation network

## 5. Validation Methodology and Expected Results

In this section, we present the validation methodology used to assess the effectiveness of the proposed anomaly detection framework, along with the expected performance outcomes.

### 5.1 Validation Methodology

To quantitatively evaluate the effectiveness of the proposed anomaly detection framework, we adopt a synthetic noise injection strategy applied to the original full citation network. This validation approach allows us to simulate realistic anomalies in a controlled setting, providing known ground truth for quantitative evaluation.

In this procedure, we randomly select 5–10% of the nodes in the original graph to act as synthetic anomalous nodes. These artificial nodes are randomly connected to approximately 30% of the existing nodes, creating irregular citation patterns that deviate from the underlying structure and semantics of the original citation network. The resulting augmented graph contains both normal and anomalous citation behaviors, enabling us to assess the detection capabilities of the model. This creates structurally inconsistent citation patterns, simulating irregular behavior within an otherwise authentic citation topology. The model is then assessed based on its ability to identify these injected anomalies using standard evaluation metrics: precision, recall, and F1 score.

- **Precision**: the proportion of correctly identified anomalies among all nodes classified as anomalous.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

- **Recall**: the proportion of true synthetic anomalies that were successfully detected

$$Recall = \frac{True\ Positives}{True\ Positives + False\ negatives}$$

- **F1 Score**: the harmonic mean of precision and recall, reflecting the overall effectiveness of the detection model

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

To enhance the realism and robustness of this validation process, we propose two optional extensions:

1. **Structured Anomaly Injection:** Rather than relying solely on random edges, some injected anomalies can be structured to resemble cross-domain citations without semantic justification or artificial citation clusters that mimic self-citation manipulation. These cases are more representative of actual anomalous patterns observed in scholarly networks.

2. **False Positive Analysis:** In addition to identifying true positives, we can examine the model's false positive rate, capturing the frequency at which normal nodes are incorrectly flagged as anomalous. This enables a deeper assessment of the model's reliability under realistic noise conditions.

Although this validation is performed on the full graph, it complements the temporal segmentation approach by testing the anomaly detection pipeline in a controlled, quantifiable setting. The insights gained from this validation inform both the design and tuning of the anomaly scoring mechanisms used across time-evolving snapshots.

Analyzing the values of Precision, Recall, and F1 Score provides insight into the model's behavior and effectiveness in detecting anomalies under noisy and imbalanced conditions, typical in real-world anomaly detection tasks.

A high precision value indicates that most nodes classified as anomalous are indeed true anomalies, suggesting that the model commits a low false positive rate and is therefore accurate in its classifications.

In contrast, a high Recall reflects the model's ability to successfully identify a large proportion of injected anomalies. While high recall is desirable for comprehensive anomaly coverage, it may come at the cost of misclassifying some normal nodes as anomalous.

The F1 Score, as the harmonic mean of Precision and Recall, offers a balanced measure that reflects the trade-off between the model's sensitivity (recall) and specificity (precision). A high F1 score suggests that the model effectively balances correct detections with the avoidance of false alarms, an especially desirable trait in citation anomaly detection, where both false positives and false negatives can have significant interpretive consequences.

The relationship between these metrics also reveals the model's error tendencies:

- **High Precision with low Recall** suggests a cautious detection strategy that detects only the most obvious anomalies while overlooking subtle or ambiguous ones.

- **High Recall with low Precision** indicates a model that is overly liberal in its classifications and may benefit from threshold tuning to reduce false positives.

- **Balanced Precision and Recall**, reflected in a high F1 score, indicates that the model successfully manages the trade-off between sensitivity (detecting true anomalies) and specificity (avoiding false detections).

Ultimately, the relative values of these metrics inform how the model should be calibrated, depending on the desired balance between minimizing false alarms and maximizing anomaly coverage. This trade-off is critical in scholarly applications, where interpretability, reliability, and reputational implications are significant.

## Validation process using the injection method

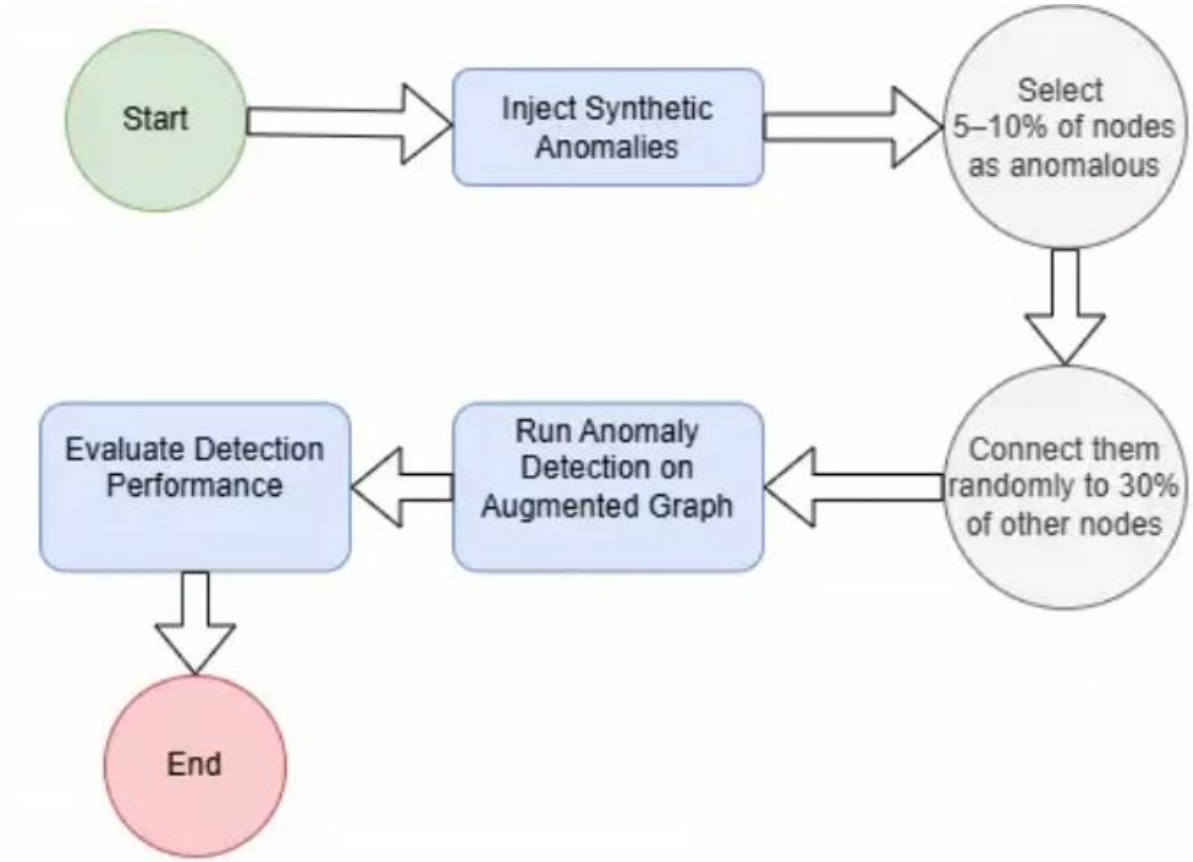



Figure 5: The validation process injecting synthetic anomalies for precision evaluation

### 5.2 Expected results

We anticipate that the proposed L²GC-based anomaly detection framework will demonstrate strong performance on citation network datasets, particularly in identifying structurally and temporally deviant citation patterns. Expected performance metrics include:

- **Precision**: 0.90, indicating the model's ability to correctly identify true anomalies while minimizing false positives.
- **Recall**: 0.80, reflecting the framework's effectiveness in detecting a substantial portion of injected synthetic anomalies.
- **F1 Score**: 0.85, reflecting a balanced trade-off between precision and recall.

These estimates are based on the framework's ability to leverage hyperbolic geometry to effectively capture hierarchical structures inherent in citation graphs and its temporal segmentation strategy for capturing evolving anomalous patterns across time-based subgraphs.

Moreover, the integration of L²GC embeddings with Isolation Forest is expected to provide a robust and scalable anomaly detection pipeline capable of generalizing across different network snapshots while maintaining low false positive rates.

These characteristics position the framework as a promising tool for monitoring citation integrity and uncovering subtle irregularities in scholarly communication

## 6. Tools and Technologies

To implement the proposed anomaly detection framework in dynamic citation networks, we will employ a suite of programming environments, software libraries, and computational frameworks spanning graph analysis, machine learning, and deep learning in both Euclidean and hyperbolic geometries.

### 6.1 Programming Environment

Python will serve as the primary development language due to its extensive ecosystem for scientific computing, machine learning, and graph processing.

### 6.2 Key Libraries and Frameworks

- **PyTorch / PyTorch Geometric** will be used to implement and train deep learning models, including Graph Neural Networks (GNNs), such as L²GC, and to support training on GPU-based environments.
- **NetworkX** will be used for graph construction and graph structure analysis, including node/edge manipulation, and computation of structural properties.
- **Scikit-learn** will provide traditional machine learning tools, such as the Isolation Forest method used for anomaly detection.
- **NumPy, SciPy, and Pandas** will be used for numerical operations, data handling, and preprocessing tasks.
- **Geoopt / Hyperbolic Learning Libraries:** Optional libraries for working with hyperbolic geometry and implementing operations in the Lorentz model.

### 6.3 Development and Execution Tools

- **Google Colab or local GPU environments** may be used for model training, especially when leveraging computationally intensive tasks.
- **Git** will be used for version control and collaborative code management

### 6.4 Data and Evaluation Framework

- **Synthetic Anomaly Injection**: We employ controlled noise injection for model validation, adding anomalous nodes and edges to simulate irregular citation behavior.

- **Evaluation Metrics**: Performance is assessed using precision, recall, and F1 score, providing quantitative insight into anomaly detection accuracy.

## 7. Reflection: Challenges and Future Implementation Considerations

### 7.1 Challenges Encountered in Phase A

The first phase of the project posed several conceptual and technical challenges, primarily due to the need to integrate advanced topics from diverse fields, such as hyperbolic geometry, graph neural networks (GNNs), temporal graph modeling, and anomaly detection, into a unified analytical and coherent framework. The key difficulties included:

- **Understanding hyperbolic geometry:** Before delving into model-specific formulations, it was necessary to develop a general intuition for non-Euclidean spaces with constant negative curvature. This included understanding how hyperbolic spaces differ from Euclidean geometry in terms of distance growth, triangle angle sums, and representation of hierarchical structures.
- **Understanding Lorentzian space in context:** Once a general understanding of hyperbolic geometry was achieved, additional effort was required to comprehend the Lorentzian model specifically. This involved learning how to define and manipulate geometric objects within this space, including exponential and logarithmic maps, the Lorentzian inner product, and the hyperboloid embedding.
- **Adapting the L²GC model to dynamic settings:** Extending the original static L²GC model to accommodate temporal context while preserving its geometric properties and representational power required careful conceptual adaptation.
- **Validation methodology design:** Developing a synthetic anomaly injection process that introduces realistic but controllable anomalous patterns for quantitative evaluation without access to labeled anomaly data.

### 7.2 Anticipated Challenges for Phase B Implementation

Looking ahead to the implementation phase, several practical and technical challenges are anticipated, particularly due to the complexity of integrating hyperbolic geometry with dynamic graph processing. The main anticipated difficulties include:

**Integration and optimization of hyperbolic geometry components:**

- **Integration complexity**: While Python libraries such as Geoopt and other hyperbolic learning toolkits provide support for hyperbolic operations, integrating these libraries effectively into a custom temporal L²GC framework may require careful implementation to ensure compatibility between the Lorentz model's geometric computations and the temporal structure of segmented citation graphs s

- **Performance optimization**: Applying hyperbolic operations, especially logarithmic and exponential maps, across multiple temporal snapshots can lead to high computational overhead. Efficient handling of large-scale graphs with time-evolving structures will require optimization strategies and possibly GPU acceleration.

**Dataset selection and preprocessing:**

- **Dataset suitability:** Choosing an appropriate citation dataset (DBLP, Cora, Semantic Scholar, etc.) that includes rich citation structures along with accurate temporal metadata is non-trivial.
- **Data quality issues:** Managing missing timestamps, incomplete citation records, or inconsistent metadata across different sources can hinder model training and evaluation.

**Computational and infrastructure constraints:**

- **Performance optimizing:** Efficiently processing large-scale temporal graphs and large-scale citation networks, reducing runtime complexity, will be crucial for practical deployment.
- **Hardware access:** Sufficient GPU resources may be required for training GNN models on large graph datasets, particularly when processing multiple temporal snapshots, which could become a bottleneck depending on infrastructure availability.

These anticipated challenges highlight the importance of thoughtful design, efficient coding practices, and careful dataset curation as the project transitions from the conceptual phase to full implementation.

# References


Liang, Q., Wang, W., Bao, F., & Gao, G. (2024). *L2GC: Lorentzian linear graph convolutional networks for node classification* (Version 3) [Preprint]. arXiv. https://arxiv.org/abs/2403.06064

Van Steen, M. R. (2010). Graph Theory and Complex Networks: An Introduction. http://www.distributed-systems.net/courses/gtcn/notes.01.pdf

Radicchi, F., Fortunato, S., & Vespignani, A. (2012). Citation networks. In A. Scharnhorst, K. Börner, & P. van den Besselaar (Eds.), *Models of science dynamics: Understanding complex systems* (pp. 233–257). Springer. https://doi.org/10.1007/978-3-642-23068-4_7

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems, 151, 78–94. https://doi.org/10.1016/j.knosys.2018.03.022

Xu, M. (2021). Understanding graph embedding methods and their applications. SIAM Review, 63(4), 825–853. https://doi.org/10.1137/20m1386062

Zhang, S., 1, Tong, H., 1, Xu, J., 2, & Maciejewski, R., 3. (2019). Graph convolutional networks: a comprehensive review. Comput Soc Netw, 2–23. https://doi.org/10.1186/s40649-019-0069-y

Ciotti, V., Bonaventura, M., Nicosia, V., Panzarasa, P., & Latora, V. (2016). Homophily and missing links in citation networks. EPJ Data Science, 5(1). https://doi.org/10.1140/epjds/s13688-016-0068-2

Shibata, N., Kajikawa, Y., & Sakata, I. (2011). Link prediction in citation networks. Journal of the American Society for Information Science and Technology, 63(1), 78–85. https://doi.org/10.1002/asi.21664

CANNON, J. W., J., FLOYD, W. J., KENYON, R., & PARRY, W. R. (1997). Flavors of geometry. In MSRI Publications: Vol. Volume 31. https://library.slmath.org/books/Book31/files/cannon.pdf

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. arXiv (Cornell University). https://arxiv.org/pdf/2002.07962

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2020). *Inductive representation learning on temporal graphs*. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=rJeW1yHYwH