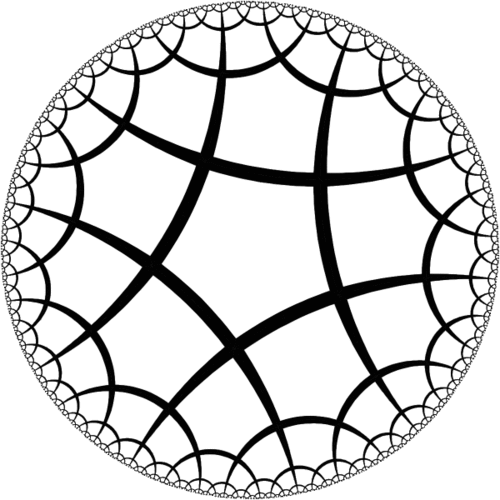
****

**Software Engineering Department**

Capstone Project Phase A

**Citation Anomaly Detection via Lorentzian Embeddings and Temporal Attention**

**25-2-R-3**

****

**Nadir Yaakov - 325173433**

**Ben Zakai - 207978545**

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

[**GitHub**](https://github.com/goodpvp90/FinalProject)

**Contents**

[Abstract 3](#_qfpc8l74rze7)

[**1. Introduction 3**](#_3x7tpkdhj4d0)

[**2. Theoretical Background 5**](#_ay8m7ykpfl3d)

[2.1 Graph Theory and Networks 5](#_6qaffncyf2f9)

[2.1.1 Citation Networks 5](#_g8fylruo5lj2)

[2.2 Graph Embedding 6](#_v0dqiyrha07l)

[2.2.1 Embedding in Static Networks 7](#_7bzcu4gexijd)

[2.2.2 Embedding in Dynamic Networks 7](#_rylga0dqq3c2)

[2.3 Graph Neural Networks (GNNs) 8](#_sad0v7rsoxy4)

[2.4 Hyperbolic Geometry 9](#_wahb5ammx21w)

[2.5 Anomaly detection 10](#_hhj9xhkx3pqs)

[**3. Preliminaries 10**](#_8685vzqvaycd)

[3.1 The Lorentz Model of Hyperbolic Geometry 10](#_1uwb0ihgymd1)

[3.2 The L2GC Model 12](#_yfmncsibpx9w)

[3.2.1 Parameter-Free Neighborhood Feature Propagation in Euclidean Space 12](#_80ihgus70to)

[3.2.2 Lorentzian Linear Feature Transformation in Hyperbolic Space 13](#_mpp07ey0lsi)

[3.3 Node labels prediction in Euclidean space 14](#_slblogmjf3e2)

[**4. The Approach 14**](#_njaxyj38f2mi)

[4.1 Input: Citation Network 14](#_q775us4cclhh)

[4.2 Data Preparation and Temporal Segmentation 15](#_86ja3liiqqm1)

[4.3 L²GC Model on Temporal Subgraphs 15](#_ljbhorp6qvhf)

[4.3 Tracking Temporal Feature Evolution 16](#_exrrz8t3xkpb)

[4.4 Anomaly Scoring with L²GC 17](#_nngg1l6oukk9)

[**5. Validation Methodology and expected Results 17**](#_lwdu67rb53po)

[5.1 Validation Methodology 17](#_ns1kmmqzxqbg)

[5.2 Expected results 19](#_5beop0xoo6q2)

[**6. Tools and Technologies 20**](#_31z9l0lnyl7v)

[6.1 Programming Environment 20](#_9jgy5e72hdt6)

[6.2 Key Libraries and Frameworks 20](#_ko75c46rxqcf)

[6.3 Development and Execution Tools 20](#_h279ofr6mvz)

[6.4 Data and Evaluation Framework 21](#_dkeorkpcyf5x)

[**7. Reflection: Challenges and Future Implementation Considerations 21**](#_f1w7tkf65pmu)

[7.1 Challenges Encountered in Phase A 21](#_cohxdiwg884c)

[7.2 Anticipated Challenges for Phase B Implementation 21](#_6kio9teo3hss)

## **Abstract**

Citation networks provide valuable insight into the evolution of knowledge through scholarly publications, where nodes represent academic papers and edges denote citations relationships. Anomalies within these 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 an anomaly detection framework that extends the Lorentzian Linear Graph Convolutional Network (L²GC) to dynamic citation networks. L²GC 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 L²GC 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 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 that characterize 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). In particular, 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 (L²GC), 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 new framework for anomaly detection in dynamic citation networks by extending the L²GC model to a temporal setting. Specifically, the approach suggests segmenting the full citation network into a series of time-based snapshots (e.g., by year), apply the L²GC model independently to each temporal subgraph, and track 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.

# 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. 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 authorsandedge**s** indicate citations**.** A directed edgepoints 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 modelsprovides 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, embeddings enable deeper understanding of network properties, enhance interpretability, and improve 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 analysing 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 node's 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 a number of 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.

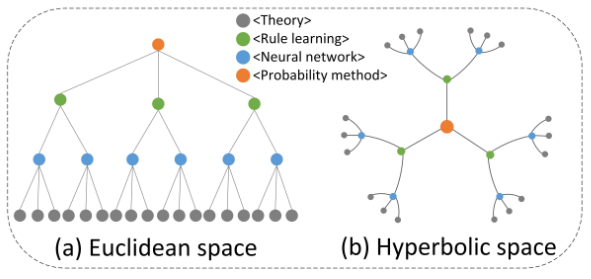


Figure 1 (Liang, Wang, Bao, & Gao, 2024): Visualization of a tree-like hierarchical structure citation network in Euclidean and hyperbolic space. In both graphs, nodes represent documents, and edges represent citations. Each color node represents a type of document.

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-likestructures [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 oflinear 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 𝐿ₖⁿ = (𝐿ⁿ, 𝑔ₖˣ), where 𝐿ⁿ is the upper sheet of a two-sheeted hyperboloid in Minkowski space and 𝑔ₖˣ denotes the Riemannian metric tensor given in this model by 𝑔ₖˣ = diag(−1, 1, …, 1).

The point set of the Lorentz model is defined as:

𝐿ⁿ = {𝑥 ∈ ℝⁿ⁺¹ | ⟨𝑥, 𝑦 = ¹⁄ₖ, 𝑥₀ > 0}

where ⟨, is the Lorentzian Inner Product, defined as c

for any two points 𝑥 = (𝑥₀, …, 𝑥ₙ) and 𝑦 = (𝑦₀, …, 𝑦ₙ) in the Lorentz space, as:

⟨𝑥, 𝑦 = −𝑥₀𝑦₀ + ∑ⁿᵢ₌₁ 𝑥ᵢ𝑦ᵢ

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.

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.

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

* **Exponential Map**: Given a tangent vector , the exponential map at point *x* maps it to the hyperbolic space 𝐿ₖⁿ along a geodesic and is defined as:

ex(𝑧) = cosh(α)𝑥 + sinh(α),

where α = and = .

* **Logarithmic Map:** The inverse mapping from a point in the hyperbolic space back to the tangent space 𝑇ₓ𝐿ₖⁿ at x is given by:

logₖˣ(𝑦) = · (𝑦 − *β*𝑥),

where *β* = 𝑘⟨𝑥, 𝑦.

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, and ​where *k* < 0 denotes the constant negative curvature, can be achieved using exponential and logarithmic maps centered at the origin. Given a function f: ℝⁿ → ℝᵐ, its canonical Lorentzian extension to the hyperbolic setting, denoted as , is defined as:

(x) = ( ( (x) ) )

Here, x ∈ is a point in the source hyperbolic space, and (x) ∈ maps this point to the tangent space at the origin. The intermediate function  applies the Euclidean transformation 𝑓̂ only to the spatial coordinates (*v*₁, ..., *vₙ*) while setting the temporal component *v*₀ = 0. That is:

(v) = (0, *f*(*v*₁, ..., *vₙ*))

Finally, the exponential map maps the transformed point back to the target hyperbolic space .

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 g 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 *G̃* is created by adding self-connections to the original graph *G*.
* A linear propagation matrix *P* is computed as

*P* = (*D* + *I*)⁻¹ᐟ² (*A* + *I*) (*D* + *I*)⁻¹ᐟ²,

where *A* is the adjacency matrix and *D* is the degree matrix of *G̃*.

* Node features are updated using the rule:

*H*⁽*ˡ*⁺¹⁾ = (1 − α) *P H*⁽*ˡ*⁾ + *αX*,

where *X* is the initial node feature matrix, and 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.

* After *n* iterations, the final node features are obtained.

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.

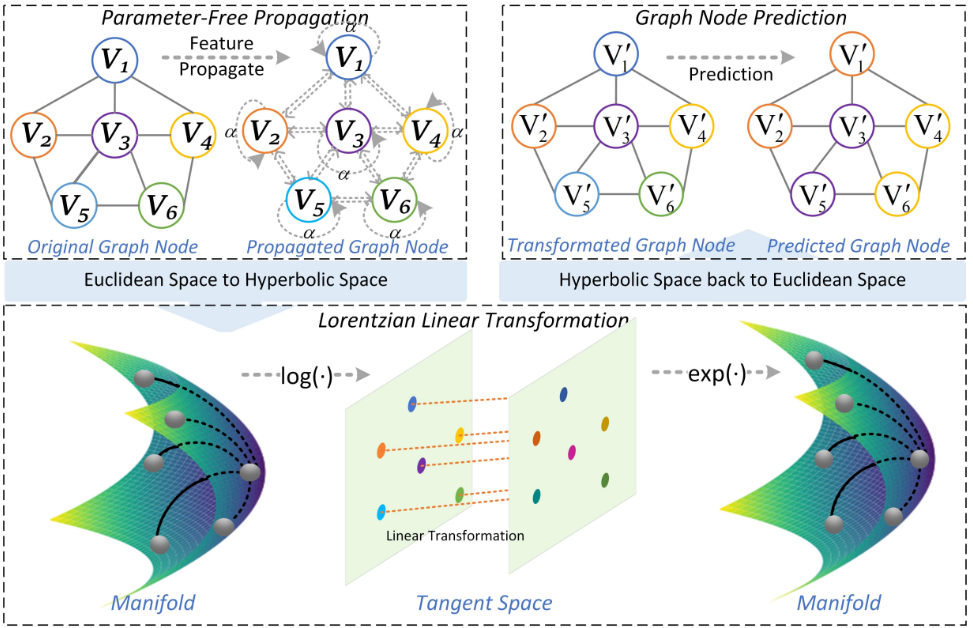


Figure 2 (Liang, Wang, Bao, & Gao, 2024): The framework of Lorentzian Linear Graph Convolutional Networks for node Classification.

### 3.2.2 Lorentzian Linear Feature Transformation in Hyperbolic Space

After propagation, the learned Euclidean node features are mapped to the Lorentz model of hyperbolic space using the exponential map at the origin, denoted *expₖ*⁰. This yields the hyperbolic embedding:

Once in hyperbolic space, L2GC performs a Lorentzian linear transformation, which mimics the role of a linear layer in Euclidean GCNs but respects the geometry of the hyperbolic manifold.

This transformation proceeds in the following sub-steps:

* Logarithmic map: The node embeddings ​ are mapped from the Lorentz manifold back to the tangent space at the origin:
* Linear transformation in tangent space: A Euclidean linear transformation is applied to the spatial components (𝑣₁, …, 𝑣ₙ), omitting the time-like component:

(𝑣) = (0, 𝑀(𝑣₁, …, 𝑣ₙ))

where is a learnable weight matrix. This respects the Lorentz structure by not transforming the first coordinate.

* Exponential map:The transformed tangent vectors are then mapped back to the Lorentz manifold:

(x)= expₖ⁰((logₖ⁰(𝑥)))

This process ensures that all operations respect the curvature and structure of the Lorentzian space while still allowing for linear transformations analogous to those in classical neural networks.

## **3.3 Node labels prediction in Euclidean space**

In the final stage of the L2GC framework, the transformed embeddings in the hyperbolic space are mapped back to Euclidean space using the logarithmic map

This mapping allows the application of standard classifiers (e.g., SoftMax layers)for node classification. By returning to Euclidean space, the model ensures compatibility with common loss functions and evaluation protocols.

By performing linear feature transformation in the Lorentz 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 -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, we extend its application to dynamic citation networks by dividing the graph into a sequence of temporal subgraphs and applying the model independently to each. This temporal decomposition allows the model to capture time-dependent structural changes and evolving citation behaviors, capabilities not available in the original, static formulation of L²GC. By combining temporal resolution with hyperbolic embedding, this approach aims to improve the detection of anomalies that emerge over time in citation dynamics.

## 4.1 Input: Citation Network

We model the academic citation network as a *temporal graph*, where represents academic papers (nodes) and 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

To capture the above-mentioned evolving structure, we partition the citation timeline into sequential intervals (e.g., yearly or multi-year windows) and construct a series of temporal subgraphs , corresponding to the citation network at a specific time window . Each subgraph contains all nodes and edges observed within that time window. This segmentation preserves the chronological order of citations and exposes the growth of the network over time. Unlike a single static graph, the temporal representation explicitly models how citation relationships form and change.

* Temporal edges: Each node corresponds to a paper, and each directed edge represents a citation occurring at a specific time.
* Time windows: The citation history is split into consecutive time intervals (for example, by year) to create a sequence of graph snapshots. This approach captures the network’s evolving topology.
* Evolving structure: By comparing these snapshots, one can highlight anomalies as unexpected shifts between intervals.

This temporal segmentation makes the dynamic nature of the citation graph explicit and enables our models to detect changes that would be hidden in a static, aggregated view/

## 4.3 L²GC Model on Temporal Subgraphs

The L²GC model (Lorentzian Linear Graph Convolution) is applied separately to each temporal subgraph ​ to learn hyperbolic embeddings of the nodes.

For each temporal subgraph , we apply the L²GC model to learn hyperbolic embeddings of the nodes. The embedding process begins by assigning to each node a static feature vector , representing attributes such as publication metadata or citation statistics. Since these feature vectors do not encode dynamic or time-dependent information, we perform Euclidean embedding once at the beginning of the process, using methods such as Node2Vec. The resulting feature matrix is then reused across all temporal subgraphs as input to the L²GC model.

The L²GC model maps the Euclidean embeddings into the Lorentzian hyperbolic space by first projecting them to the tangent space at the origin and then performing Lorentzian graph convolutions. This transformation enables the model to capture hierarchical and tree-like patterns that are present in citation networks. The resulting hyperbolic embeddings represent each node's position within the structure of the subgraph, incorporating both local and global topological features. This approach reduces computational overhead and ensures consistency of node representations across time, allowing the model to focus on capturing structural evolution rather than redundant feature transformations.

To summarize, the four main steps at this stage are as follows:

* Euclidean embedding: We begin by projecting the static feature vectors of each node (e.g., metadata or citation counts) into a lower-dimensional Euclidean space using techniques such as Node2Vec. This step is performed once and reused across all temporal subgraphs, as the feature vectors are time-invariantץ
* Hyperbolic transformation: In each snapshot, we project the Euclidean node features into hyperbolic space. L²GC then performs a Lorentzian linear transformation, which is adept at capturing the citation network’s hierarchical patterns.
* Graph convolution: L²GC aggregates information from a node’s neighbors in the Lorentzian space, similar to a graph convolution but tailored to hyperbolic geometry.
* Temporal application**:** Unlike prior work, where L²GC was evaluated on a single static citation graph, we run L²GC on each time-segment subgraph independently. This produces a distinct embedding for each node (or edge) in each time window, effectively “re-using” L²GC over time.

By applying L²GC on each temporal subgraph, we leverage its representational power while adapting it to the dynamic setting. The sequence of L²GC embeddings then forms the basis for analyzing how citation patterns evolve over time.

## 4.3 Tracking Temporal Feature Evolution

After computing L²GC embeddings for each time window, we obtain a trajectory of features for each node across intervals. We then analyze these trajectories to detect abnormal changes. Specifically, for each node, we compare its embedding at time with its embedding at time . A *normal* evolving pattern corresponds to smooth, gradual changes in embedding space, while a sudden jump or drift indicates a potential anomaly. This approach is analogous to methods in temporal GNNs that extract features from each snapshot and capture their evolution.

In brief, the three main steps at this stage include:

* Sequential embeddings: For each time index , let be the matrix of L²GC node embeddings on the subgraph ​. We form a time series for the entire graph.
* Change detection: We measure how much each node’s embedding changes between ​ and ​. Large changes (e.g., large Euclidean distance in embedding space) signal an unexpected shift. Such deviations often correspond to anomalous citation behavior, as noted in temporal graph theory.
* Pattern analysis: Tracking these embedding trajectories 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.

In summary, by segmenting the graph and comparing L²GC outputs over time, we can pinpoint when and where the network’s behavior diverges from the norm.

## 4.4 Anomaly Scoring with L²GC

For each temporal subgraph, we apply the Isolation Forest (IForest) algorithm to the node embeddings obtained from L²GC after mapping them back to Euclidean space via the logarithmic map. IForest assigns an anomaly score to each node , indicating its deviation from typical citation patterns within that subgraph.

By collecting these scores over all time windows, we construct a temporal anomaly vector

for each node . This vector captures the evolution of the node’s anomaly behavior over time.

We use this vector to detect anomalous nodes by:

* 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.

This temporal approach enhances the detection of evolving and context-sensitive anomalies in citation networks.

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

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

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.

Interpreting Evaluation Metrics:

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

## 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**: 0.80-0.85, indicating the model’s ability to correctly identify true anomalies while minimizing false positives.
* **Recall**: 0.75-0.80, reflecting the framework’s effectiveness in detecting a substantial portion of injected synthetic anomalies.
* **F1 Score**: 0.77-0.82, 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.
* **Temporal segmentation strategy:** Designing a segmentation approach that balances temporal resolution with computational feasibility, especially when dealing with large-scale citation networks.
* **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 include 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 GNNs 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>