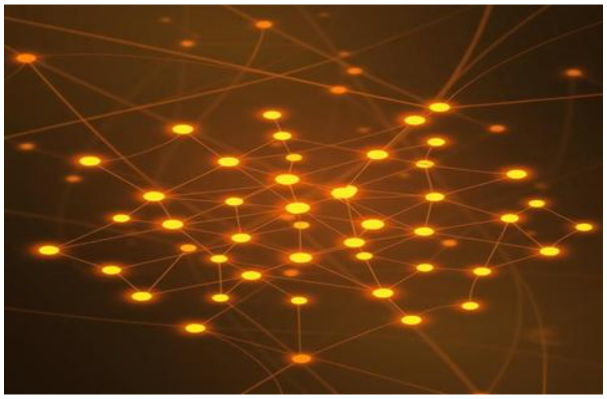
Software Engineering Department

Braude College of Engineering

Capstone Project Phase B

25-1-R-3

Detecting Citation Anomalies in Hyperbolic Space with the Poincaré Ball Model



Yotam Gilad

Tomer Ben Lolo

Supervisor: Dr. Dvora Toledano Kitai

https://github.com/YotamG12/1.-Analyzing-Dynamic-neural-network-graph-within-Hyperbolic-Space.git

Abstract 4
1. Introduction 4
2. Theoretical Background 6
2.1 Embeddings in Static Networks 6
2.1.1 Embedding in Simple Static Networks 6
2.1.1.1 DeepWalk- 6
2.1.1.2 Node2vec- 7
2.1.2 Euclidean Embedding in Complex Static Networks 7
2.1.2.1- Graph Attention Networks 7
2.1.2.2 Graph Convolutional Networks 7
2.1.3 Hyperbolic Space 8
2.1.4 Hyperbolic Embedding in Complex Static Networks 9
2.1.4.1 HGCN 9
2.1.4.2 EvolvedGCN 9
2.2 Dynamic Networks 9
2.2.1 Embedding in Dynamic Networks 10
2.2.1.1 DynamicTriad - 10
2.2.1.2 DySAT - 10
2.2.2 Dynamic Networks in hyperbolic space 10
2.3 Anomaly Detection in Citation Networks 11
2.3.1 Isolation Forest 12
2.4 Attention Mechanism 12
3. Preliminary 13
3.1 Problem Formulation 13
3.2 Hyperbolic Geometry 14
15
3.3 Feature Maps 16
3.4 Sleeping Beauty- 18
3.5 Falling Star- 19
4. Architecture of the Citation Anomaly Detection Framework 19
4.0 Preparing the Data 19
4.1 Hyperbolic Structural Attention (HSA) 20
4.1.1 Feature Vector Construction 20
4.1.2 Embedding Feature Vectors in Hyperbolic Space **שגיאה! הסימניה אינה מוגדרת.**
4.1.3 Transformation into Higher Representation 22
4.1.4 Hyperbolic Self-Attention Mechanism 24
4.1.4.1 Attention Score 24
4.1.5 Data Aggregation 25
4.1.6 Transition to Euclidean Space 25
4.2 Position Embedding 26
4.3 Anomaly Detection via Isolation Forest (IF) 26

4.3.1 Preparing the Input 26
4.3.2 Model Training and Anomaly score 27
4.3.3 Result Analysis 27
1. Representation of Anomaly Scores 27
3. Normalization 28
4. Visualization of Anomaly Scores 28
5. Temporal Change Analysis 28
6. Identification of Extreme Cases 28
5. Research Flow: 29
5.1 Phase A: Theoretical and Methodological Foundations 29
5.1.1 Literature Review and Conceptual Understanding 29
5.1.2 Methodological Framework Design 30
5.1.3 Validation Strategy and Experimental Design 30
5.2. Phase B: Implementation and Data Processing 30
5.2.1 Model Adaptation 31
5.2.2 Dataset Construction 31
5.2.3 Model Validation 31
5.2.3.1 Parameter Selection Rationale for Noise Injection Validation 31
6. Tools & softwares used 33
1. Scientific Computing 33
2. Machine Learning 33
4. Graph Neural Networks (PyTorch & PyG) 34
5. Node Embeddings 35
7. Results and Discussion 35
7.1 Temporal Anomaly Patterns: Sleeping Beauty 35
7.2 . Falling Star Pattern Analysis via ∆ Anomaly Score (ΔAS) **שגיאה! הסימניה אינה מוגדרת.**
Time Window: [10–20]: 36
7.3 Mean Anomaly Score Distribution: Evidence of Dual Behavioral Regimes 39
7.4 Interpretation Beyond Zipfian Behavior 39
7.5 Broader Implications 40
8. Challenges We Faced: 41
Bibliography 43

General Description

Abstract

This project aims to enhance the integrity and reliability of scientific communication by detecting citation anomalies through hyperbolic geometry, specifically leveraging the Poincaré ball model. Citation anomalies, such as misrepresented or irrelevant references, can undermine the credibility of scholarly work. Traditional citation analysis methods are both labor-intensive and susceptible to biases, underscoring the need for scalable automated solutions.

To address this challenge, this project introduces an approach that utilizes hyperbolic embeddings to capture the hierarchical structures of citation networks. These embeddings are particularly effective in modeling the exponential growth of relationships in scale-free networks, which characterize academic citation graphs. The proposed model, inspired by the DynHAT framework, comprises three key components: Hyperbolic Structural Attention (HSA), Position Embedding, and Anomaly Detection using Isolation Forest. HSA extracts node features and embeds them in hyperbolic space, employing attention mechanisms to emphasize semantically significant connections. Position embeddings incorporate temporal context, enabling the model to track shifts in citation behavior over time. Finally, the anomaly detection stage identifies papers exhibiting irregular citation patterns, such as sudden surges or drops in influence.

The model's performance is assessed by introducing controlled perturbations into the citation network to evaluate its anomaly detection capabilities. Results show that the model successfully identifies approximately 91.8% of injected anomalies, demonstrating the effectiveness of the proposed framework. By integrating hyperbolic geometry with dynamic graph analysis, this research provides a scalable and accurate method for detecting citation misuse, thereby improving the evaluation and trustworthiness of scholarly communication.

1. Introduction

Within the interconnected network of scientific literature, citations served as the foundation of scholarly communication, enabling the validation of ideas, the contextualization of findings, and the tracing of knowledge development. However, citations were not universally reliable. Some were misrepresented, taken out of context, or rendered inaccurate. These citation anomalies, whether intentional or inadvertent, distorted the scientific narrative and undermined the credibility of the academic ecosystem.

Detecting and addressing citation anomalies was a challenging task, particularly given the exponential growth in the volume of published research. Traditional methods for assessing citation accuracy relied primarily on manual review, a resource-intensive process that was also vulnerable to subjective biases. Recent advancements in computational models, particularly in the domain of natural language processing (NLP) and machine learning, presented a promising pathway for automating the detection of citation anomalies, improving both precision and scalability.

This work introduced an innovative approach for detecting citation anomalies using embeddings in hyperbolic space, specifically utilizing the Poincaré ball model. Unlike traditional Euclidean embeddings, hyperbolic ones proved particularly effective for modeling hierarchical and relational data, such as citation networks. By embedding scientific articles, their citations, and associated textual content into hyperbolic space, we aimed to capture subtle relationships and identify inconsistencies that might have indicated citation misuse or misrepresentation.

Through this innovative application of hyperbolic geometry, our approach sought to address the following objectives:

- To model the hierarchical structure of citation networks efficiently using hyperbolic space.
- To detect semantic and contextual discrepancies in cited text by comparing them with their usage in citing articles.
- To develop a robust algorithm capable of scaling large datasets while maintaining high accuracy in anomaly detection.

By using the framework of the Poincaré ball model, we aimed to advance the field of citation analysis and contribute to the broader objective of enhancing the integrity and reliability of scientific communication. This project outlined the theoretical foundation of our approach, provided a detailed description of the methodology, and discussed the implications of detecting citation anomalies using hyperbolic space representations.

The integration of hyperbolic geometry with citation networks offered a promising approach to uncovering latent hierarchical relationships, detecting anomalies, and deepening the understanding of complex interconnections within scientific literature. Recent studies explored this intersection, addressing critical challenges in network analysis.

The study [3] introduced a dynamic graph embedding model designed to capture both structural and temporal dynamics of evolving graphs, such as social or communication networks. The proposed approach leveraged the Lorentz space for dynamic graph embedding, combining a structural layer that mapped graph topology into Lorentz space with a temporal layer that employed self-attention mechanisms to model time-evolving dependencies. A key advantage of this method lay in its ability to effectively balance structural proximity with temporal dynamics.

The study [1] investigated the potential of hyperbolic geometry to enhance the optimization and performance of Binary Neural Networks (BNNs). The authors introduced the Hyperbolic Binary Neural Network (HBNN), which leveraged the Riemannian exponential map to reformulate the constrained optimization problem in hyperbolic space as an unconstrained problem in Euclidean space. Furthermore, they proposed the Exponential Parametrization Cluster (EPC) method, which employed a diffeomorphism to contract the segment domain, thereby increasing the likelihood of weight flips and maximizing gain in BNNs. The hyperbolic framework significantly improved the network's generalization capabilities, particularly for tasks involving hierarchical or complex relationships.

2. Theoretical Background

The study of citation anomalies and the structural analysis of citation networks intersect with a range of fields, including graph theory, network science, and computational linguistics. To properly contextualize the use of the Poincaré ball model in hyperbolic space for detecting citation anomalies, it is crucial to understand the evolution of network analysis methodologies and their relevance to this area of study.

2.1 Embeddings in Static Networks

2.1.1 Embedding in Simple Static Networks

Simple static networks represent relationships in a fixed, unchanging structure, often modeled using graph representations as adjacency matrices or edge lists. These networks are fundamental for understanding pairwise connections, such as those present in co-authorship graphs or direct citation links. Early methods in citation analysis relied primarily on simple static network models, utilizing metrics such as degree centrality, clustering coefficients, and shortest paths to infer influence or detect anomalies.

Euclidean space embedding has played a foundational role in network analysis, offering a robust framework for projecting high-dimensional relationships into lower-dimensional spaces. This capability is critical for tasks such as visualization and understanding the structural properties of networks. Notable techniques including node2vec, DeepWalk, and Graph Convolutional Networks (GCNs) utilize Euclidean embeddings to perform various tasks, such as analyzing citation networks, detecting anomalies, classifying nodes, and predicting missing links. A key characteristic of Euclidean spaces is their linear expansion as the distance from a given point increases.

2.1.1.1 DeepWalk-

DeepWalk is an unsupervised learning algorithm designed to generate low-dimensional vector representations (embeddings) for nodes in a graph. By combining local and global structural information, DeepWalk employs truncated random walks to produce sequences of nodes, treating these sequences analogously to sentences in natural language processing. This enables the algorithm to capture both neighborhood-based and broader graph properties effectively.

2.1.1.2 Node2vec-

Node2vec [7] builds upon the principles of DeepWalk, enhancing the random walk strategy through a tunable trade-off between breadth-first and depth-first search. This flexibility allows Node2Vec to capture a broader range of structural patterns in the graph. Like DeepWalk, it generates low-dimensional embeddings that preserve the relational properties of nodes, making it particularly effective for downstream tasks such as classification, clustering, and link prediction.

While simple static networks are computationally efficient and relatively straightforward to analyze, they fall short in representing dynamic or hierarchical relationships that are prevalent in real-world systems, such as citation networks. These limitations become particularly pronounced when analyzing large-scale datasets, where complex interactions and temporal dynamics play a critical role in shaping the network structure.

2.1.2 Euclidean Embedding in Complex Static Networks

To overcome the limitations of simple static networks, researchers have turned to complex static networks that incorporate additional layers of information, such as weighted edges, multi-modal nodes, and hierarchical structures. In citation networks, these models facilitate more nuanced representations, such as assigning edge weights based on citation frequency or differentiating between self-citations and external references. These enhancements allow for a more detailed analysis of relationships and influence within the network, providing greater insight into the structure and dynamics of scholarly communication.

2.1.2.1- Graph Attention Networks

Graph Attention Networks (GAT) [10] utilize masked self-attentional layers to overcome the limitations of earlier approaches based on graph convolutions or their approximations. By employing a self-attention mechanism, GAT dynamically assigns weights to neighboring nodes, enabling the model to prioritize relevant connections and effectively capture complex relational patterns within the graph.
The self-attention mechanism is a sophisticated approach that our model effectively leverages to enhance its performance and representation capabilities.

2.1.2.2 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) [11] are a class of neural networks specifically designed to learn from graph-structured data, extending the concept of convolution to non-Euclidean domains. GCNs operate by aggregating information from the local neighborhood of each node, allowing nodes to iteratively update their representation based on their own features and those of their neighbors. This iterative process, known as message passing, is applied across layers, enabling the network to capture both local and global structural patterns in the graph. GCNs are highly effective for a variety of tasks such as node classification, link prediction, and graph-level classification, as they preserve the relationships and attributes inherent to the graph. By leveraging adjacency matrices and node feature vectors, GCNs efficiently encode the structures and attributes of graphs into low-dimensional representations, making them well-suited for downstream tasks that require a nuanced understanding of graph relationships and properties.

These methods assume that the representation space is Euclidean, which limits their ability to accurately model structures with hierarchical complexity or large-scale patterns, as is often the case in social networks, and many dynamic graphs.

Hyperbolic space offers a promising alternative for graph modeling, particularly in complex graphs with hierarchical structures. In recent years, an increasing number of studies have attempted to generalize graph convolution to hyperbolic space.

2.1.3 Hyperbolic Space

To effectively model structures that exhibit natural expansion, hyperbolic space, characterized by its constant negative curvature, provides a unique and robust geometric framework. This space is particularly well-suited for representing hierarchical and scale-free structures frequently observed in complex datasets such as citation networks, taxonomies, and social networks. The strength of hyperbolic space lies in its intrinsic property of exponential growth, which aligns closely with the branching patterns of hierarchical systems and networks. In such structures, relationships typically emanate from central, highly influential nodes and extend outward to peripheral, less connected ones.

Unlike Euclidean space, which faces challenges in efficiently representing growth patterns due to its linear and flat geometry, hyperbolic space offers a compact, efficient, and semantically meaningful framework for capturing these intricate relationships. This geometric advantage allows hyperbolic embedding to more effectively preserve distances, proximities, and inherent hierarchical structures within the data. Consequently, hyperbolic embedding enables more accurate modeling of subtle and complex relationships, making them highly suitable for analyzing hierarchical and scale-free systems.

2.1.4 Hyperbolic Embedding in Complex Static Networks

2.1.4.1 HGCN

HGCN [2] is a static embedding method that combines the expressive power of Graph Convolutional Networks (GCNs) with the unique properties of hyperbolic geometry to learn node representations for hierarchical and scale-free graphs. This approach leverages the advantages of both GCNs and hyperbolic space, enabling more effective modeling of complex structures inherent in such networks.

2.1.4.2 EvolvedGCN

EvolveGCN [2] is a temporal model that extends the GCN framework by computing a separate GCN model for each time step. The model is dynamically updated with each new input, utilizing a recurrent neural network (RNN), such as a Gated Recurrent Unit (GRU), to capture temporal dependencies.

The notable advantage that hyperbolic space offers for static graphs provides a strong incentive to further explore its application to dynamic graphs, where the structure evolves over time.

2.2 Dynamic Networks

Dynamic networks offer a framework for capturing the temporal evolution of relationships by modeling how connections form, dissolve, or evolve over time. In the context of citation analysis, dynamic networks are essential for tracking research trends, identifying emerging fields, and detecting citation anomalies that occur within specific timeframes.

Advanced techniques such as temporal graph networks, dynamic stochastic block models (DSBM), and time-aware node embeddings have been applied to dynamic citation networks. These methods enable the analysis of shifting influence patterns and the detection of inconsistencies. However, their reliance on Euclidean or other static spaces imposes limitations, as these spaces struggle to accurately represent complex hierarchies and hyperbolic-like structures that are intrinsic to citation data.

2.2.1 Embedding in Dynamic Networks

Embedding in dynamic networks involves learning low-dimensional representations of nodes, edges, or subgraphs in networks that evolve over time. Unlike static networks, dynamic networks account for changes in topology, node attributes, or edge weights, necessitating embeddings that adapt while preserving temporal consistency. Techniques for dynamic network embedding typically extend static methods by incorporating temporal information through recurrent neural networks (RNNs), temporal attention mechanisms, or snapshot-based models that process sequential graph states. These methods aim to capture both structural and temporal dependencies, facilitating applications such as real-time node classification, link prediction, and anomaly detection. By modeling the temporal evolution of the graph, dynamic embeddings offer a more nuanced and more flexible understanding of network behavior over time.

2.2.1.1 DynamicTriad -

DynamicTriad [2] focuses on the specific structure of triads to model how close triads are formed from open triads in dynamic networks, capturing the evolution of relationships between nodes over time.

2.2.1.2 DySAT -

DySAT [2] utilizes Graph Attention Networks (GAT) as a static layer combined with a self-attention mechanism to capture the temporal evolution of graphs. This approach allows the model to adaptively focus on relevant node interactions and track changes in the network structure over time.

2.2.2 Dynamic Networks in hyperbolic space

Dynamic networks in hyperbolic space enhance this capability by integrating temporal dynamics into the embeddings. Recent advancements, such as Hyperbolic Dynamic Graph Neural Networks (HDGNNs) and time-aware hyperbolic embeddings, have demonstrated potential in capturing the interplay between structural hierarchy and temporal evolution. By embedding nodes into hyperbolic space, these methods maintain the integrity of hierarchical relationships while adapting to changes over time, thus offering a robust framework for anomaly detection in citation networks.

Poincaré embeddings are well-suited for modeling the tree-like hierarchy in citation networks, where influential papers occupy central nodes, and less-cited works radiate outward.

In contrast to static embeddings, which assume a fixed graph structure, dynamic Poincaré embeddings are designed to adjust to temporal changes in node connections, edge weights, and the network's topology. This process often involves modeling network snapshots at different time intervals or utilizing continuous-time frameworks to capture evolving structural patterns.

2.3 Anomaly Detection in Citation Networks

Anomaly detection in citation networks entails identifying irregular patterns or unexpected behaviors, such as papers exhibiting unusual citation trends or outlier connections. Given that Citation networks are hierarchical and scale-free, they are particularly well-suited for analysis in hyperbolic space, where the exponential growth of relationships can be efficiently modeled. Anomalies in these networks may include papers experiencing sudden spikes, potentially signaling breakthrough works, or those with abnormally low citations counts, suggesting overlooked or underappreciated research. Furthermore, connections that significantly deviate from established citation patterns, such as self-citations or citations between unrelated domains, may indicate anomalous behavior. Machine learning methods, including graph neural networks (GNNs) and hyperbolic embedding such as Poincaré embeddings, are well-suited for this task as they effectively capture both the structural and temporal properties of the network. By employing these methods, researchers can uncover emerging trends, detect fraudulent citations, and identify valuable contributions that may have been overlooked, thereby enhancing the understanding and integrity of citation ecosystems. Common techniques for anomaly detection in graphs include the Local Outlier Factor (LOF), which evaluates anomaly scores by analyzing a node’s local density relative to its neighbors, and Isolation Forest, which isolates anomalies within the feature space. These methods consider both the attributes of individual nodes and their local context, making them particularly well-suited for heterogeneous networks.
The underlying rationale for these approaches is based on the observation that nodes in a well-structured graph generally conform to expected patterns, whether in terms of their features, relationships with neighboring nodes, or their role within the broader network structure. Anomalies, on the other hand, may appear as nodes with atypical attributes, unexpected connections, or behaviors that deviate significantly from their local or global context.

2.3.1 Isolation Forest

Isolation Forest [6] is a widely used anomaly detection algorithm designed to identify outliers by isolating them in the feature space. It operates on the premise that anomalies are data points that are both rare and distinct, making them easier to separate from the rest of the data compared to normal points.

The algorithm constructs a collection of random binary decision trees by recursively splitting data based on randomly chosen features and split values. Due to their distinctive nature, anomalies require fewer splits to be isolated, resulting in shorter path lengths in the decision trees. The anomaly score for each data point is computed as the average path length across all trees, where shorter paths indicate a higher likelihood of being an anomaly.

Isolation Forest is computationally efficient, scales effectively to large datasets, and does not rely on prior assumptions about the underlying data distribution. For graph-based anomaly detection, Isolation Forest can be adapted by applying it to node embeddings or local neighborhood features extracted from the graph, leveraging the graph's structural properties while maintaining the algorithm's efficiency.

In hyperbolic space, anomalies can be identified by analyzing points with unexpected radial distances, angular deviations, or atypical hierarchical positioning, assigning them higher anomaly scores to capture their divergence from expected patterns.

2.4 Attention Mechanism

Attention mechanisms [9], particularly self-attention, have transformed deep learning by empowering models to selectively focus on the most relevant parts of the input data.

The core of the attention mechanism lies in computing the relevance of different input elements, assigning weights that determine their contribution to the output.

For a given a query Q , key K , and value V, the attention score is computed as

$$\boxed{(1)\ Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_K}})V}$$

Where d_K represents the dimensionality of the key vectors.

The Softmax function [5] ensures that the attention weights are normalized, allowing the model to prioritize elements that are most relevant to the query. This mechanism serves as the foundation for many modern architectures, enabling them to dynamically capture relationships within the data.

Self-attention generalizes the attention mechanism by using the same input to derive the query (Q), key (K), and value (V) representations through learned projections. This allows the model to effectively capture intra-sequence relationships, making it particularly useful for graph-based and hierarchical data.

3. Preliminary

3.1 Problem Formulation

In this work, we formally define the problem of dynamic graph representation learning. A dynamic graph is defined as a series of observed static graph snapshots,

$G = \{G_1, \ldots, G_T\}$ where T is the number of steps in time. Each snapshot

$G_t = (V_t, A_t) \subseteq G$ is a weighted and directed network snapshot recorded at

time t ,where V_t is the set of vertices and A_t is the corresponding adjacency matrix at time step. Unlike some previous methods that assume links can only be added in dynamic graphs, we also support removal of links over time. Dynamic graph representation learning aims to learn a mapping function that obtains a low-dimensional representation for each node at time steps $t = \{1,2,\ldots,T\}$. Each node embedding h_v^t preserves both local graph structures centered at v and its temporal evolutionary behaviors such as link connection and removal up to time step t.

A notable application of this approach is the **Poincaré Embedding**, which maps hierarchical data—such as academic taxonomies, organizational charts, or citation networks—into a hyperbolic space. This method excels in preserving intrinsic relationships and latent structural patterns inherent to such datasets.

Poincaré embeddings are particularly effective in modeling research domains, where central nodes correspond to highly influential and widely cited papers, while peripheral nodes represent emerging, niche, or specialized works. By maintaining the contextual hierarchy in a natural and interpretable manner, this approach provides valuable insights into the organization and evolution of knowledge within complex networks.

3.2 Hyperbolic Geometry

Hyperbolic geometry is a branch of non-Euclidean geometry characterized by constant negative curvature which distinguishes it from the flat nature of Euclidean geometry and the positive curvature of spherical geometry. This unique curvature results in several properties that are fundamentally different from those of Euclidean such as exponential growth of distances. In hyperbolic space, distances between points grow exponentially as they move outward from the origin.

This property makes hyperbolic geometry especially suitable for representing hierarchical, tree, or scale-free structures commonly found in networks, such as citation networks. By leveraging its exponential growth property, hyperbolic space efficiently encodes multi-scale relationships while preserving both local and global structures.

A Riemannian manifold is a generalization of the concept of a curved surface to higher dimensions, providing a mathematical framework for analyzing spaces that may not adhere to the flat structure of Euclidean geometry. This space extends the concept of a two-dimensional surface to a higher dimension. At each point x on the Riemannian manifold M, there exists a **tangent space** TxM which is a Euclidean space of the same dimensionality as Mintuitively represents all possible tangential directions passing through point x (see Fig. 1).

The tangent space provides a local linear approximation of the manifold, making it possible to apply familiar Euclidean operations such as vector addition and scalar multiplication in a localized context. This is particularly useful for computations, as it simplifies working with the non-linear structure of the manifold.

In the context of hyperbolic geometry, tangent space plays a crucial role in embedding and optimization tasks, such as projecting points from the hyperbolic space to its tangent space for computations and then mapping them back to the manifold.

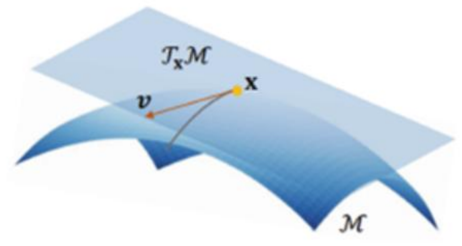


Fig.1 The tangent space $T_x M$ and a tangent vector v, along the given point x of a curve traveling through the manifold M

In hyperbolic space, the Riemannian manifold framework enables:

1. Exponential Growth Representation: the metric of hyperbolic manifolds inherently models the exponential expansion observed in hierarchical and tree-like structures.
2. Embedding Techniques: nodes or data points from hierarchical datasets, such as citation networks or taxonomies, can be embedded into hyperbolic manifolds, preserving both local neighborhoods and global hierarchical relationships.

Hyperbolic space can be described using several models, such as the Poincaré ball model, the hyperboloid model, and the Klein model, all of which are Riemannian manifolds. These models provide different coordinate systems and interpretations for hyperbolic geometry but share the same underlying structure and properties.
In this project, we will adopt the Poincaré ball model.

Poincaré Ball Model

The Poincaré model avoids the distortions associated with representing hierarchical relationships in Euclidean space. By accurately preserving the exponential growth of relationships, it allows for a more faithful representation of citation hierarchies and other scale-free structures. This capability is critical for tasks like anomaly detection, where subtle deviations in structure or position must be identified and analyzed. One of the many applications of the Poincaré ball model is the **Poincaré embedding**, which maps hierarchical data—such as academic taxonomies, organizational charts, or citation networks,into a hyperbolic space. Poincaré embeddings are particularly effective in modeling research domains, where central nodes correspond to highly influential and widely cited papers, while peripheral nodes represent emerging, niche, or specialized works.

Space Definition:

The n-dimensional Poincaré ball is defined as:

$$\boxed{(2)\ H^{n,c} = \{x \in \mathbb{R}^n |\ ||x|| < \frac{1}{\sqrt{c}}\}}$$

where $c > 0$ is the curvature parameter and $\parallel x \parallel$ is the Euclidean norm of the vector x.

The factor $\frac{1}{\sqrt{c}}$ defines the radius of the ball.

If $c = 0$, it degrades to Euclidean space, i.e., $H^{n,0} = \mathbb{R}^n$

Distance Metric:

The hyperbolic distance $d(x, y)$ between two points $x, y \in H^{n,c}$, is:

$$\boxed{(3)\ \ d(x,y) = \frac{1}{\sqrt{c}} arcosh(1 + 2\frac{||x-y||^2}{(1-c||x||^2)(1-c||y||^2)})}$$

$c > 0$: Curvature of the hyperbolic space.

$||x||$: Euclidean norm

$arcosh$ is the inverse hyperbolic cosine function.

This formula captures the hyperbolic nature of space, where distances are distorted relative to Euclidean geometry and grow exponentially as points approach the boundary of the ball $||x|| = \frac{1}{\sqrt{c}}\ or\ ||y|| = \frac{1}{\sqrt{c}}$.

The curvature parameter scales the geometry, with larger negative values of deepening the hyperbolic space and accentuating its non-Euclidean properties. This exponential growth in distances allows the metric to effectively represent hierarchical and scale-free relationships, making it particularly well-suited for structures with complex or layered connectivity.

3.3 Feature Maps

A **feature map** is a transformation process that converts raw input data into a higher-dimensional representation, enhancing its utility for tasks such as classification, regression, clustering or other predictive analyses.

By emphasizing specific patterns or properties, feature maps enable algorithms to extract and leverage meaningful insights that might remain hidden in the original data space.

The primary goal of a feature map is to project data into a more expressive feature space where relationships and patterns become more apparent. This transformation:

1. Highlights significant attributes or interactions within the data.
2. Simplifies the task of identifying correlations, clusters, or separable classes.
3. Enhances the performance of downstream algorithms by providing a representation better suited for the task at hand.

In machine learning, feature maps play a crucial role in various contexts. For example, in Convolutional Neural Networks (CNNs), feature maps represent the output of convolutional layers. Filters within these layers detect and emphasize specific patterns in the input, such as edges, textures, or shapes in images.

Feature maps are critical in graph-based learning frameworks, where they encode graph elements (nodes, edges, or subgraphs) into low-dimensional vector spaces while preserving structural and attribute information. Examples include node2vec, DeepWalk that utilize random walks to generate embeddings that capture node connectivity and positional relationships, and Graph Neural Networks (GNN's) that employ feature maps to aggregate node attributes and structural information via message-passing mechanisms, enabling tasks like node classification or link prediction.

Feature maps serve as a bridge between raw data and the task-relevant feature space, unlocking the potential for more effective learning and analysis.

Hyperbolic Embeddings and Feature Maps

Hyperbolic embeddings utilize feature maps to transform hierarchical or scale-free data into hyperbolic space. This approach provides a more efficient and compact representation of such structures compared to Euclidean space, due to the intrinsic properties of hyperbolic geometry, such as its ability to model exponential growth and hierarchical relationships.

Feature maps in hyperbolic embeddings are designed to emphasize key structural properties, such as hierarchy and scale-free patterns, which are prevalent in many complex datasets. By leveraging the unique properties of hyperbolic space, these feature maps facilitate the encoding of relationships in a manner that aligns with the intrinsic geometry of the data. This alignment enables hyperbolic embeddings to effectively capture and represent hierarchical structures (e.g., tree-like relationships) and networks with power-law distributions (e.g., scale-free graphs). Consequently, hyperbolic embeddings are particularly well-suited for tasks such as hierarchy representation, link prediction, and clustering.

By incorporating feature maps, hyperbolic embeddings achieve a powerful synthesis of structural fidelity and computational efficiency, offering a robust tool for analyzing complex data geometries.

Exponential and Logarithmic Maps

To facilitate computations and transformations between hyperbolic space and tangent space, hyperbolic embeddings rely on the exponential map and logarithmic map:

Exponential Map exp_x:

- Projects a vector from the tangent space at a reference point x back into hyperbolic space.
- This operation is crucial for transferring learned updates or features from the Euclidean tangent space into the hyperbolic manifold.

logarithmic Map log_x:

- Inverse of the exponential map, projecting points in hyperbolic space back onto the tangent space at x.
- Enables operations like gradient descent in the Euclidean tangent space, which is computationally more straightforward.

Feature maps in hyperbolic embeddings allow models to alternate seamlessly between hyperbolic and Euclidean spaces. This enables leveraging Euclidean efficiency for computation while preserving the structural integrity of hyperbolic geometry. In citation networks, this approach models the hierarchical importance of papers, positioning influential works near the origin and peripheral ones further out, reflecting their relative significance in the network.

Formally, $For\ x' \in H^{d,c}, a \in T_{x'}H^{d,c}, b \in H^{d,c}\ and\ a \neq 0, b \neq x'\ then$

Exponential **map definition:**

$$\boxed{(4)\quad exp_{x'}^{c}(a) = x' \oplus^{c} \left(tanh\left(\frac{\sqrt{c}\lambda_{x'}^{c}||a||}{2}\right)\frac{a}{\sqrt{c}\,||a||}\right)}$$

where $\lambda_{x'}^{c} = \frac{2}{1-c||x'||^2}$ is conformal factor and $\oplus$ is Mobius addition,

defined for any $u, v \in H^{d,c}$ as:

$$\boxed{(5)\quad u \oplus v = \frac{(1+2c\langle u,v\rangle + c||v||^2)u + (1-c||u||^2)v}{1+2c\langle u,v\rangle + c^2||u||^2||v||^2}}$$

logarithmic **map definition:**

$$\boxed{(6)\quad log_{x'}^{c}(b) = \frac{2}{\sqrt{c}\lambda_{x'}^{c}} artanh(\sqrt{c}||-x' \oplus^{c} b||)\frac{-x' \oplus^{c} b}{||-x' \oplus^{c} b||}}$$

Note that $x' \in H^{d,c}$ is a local reference point.

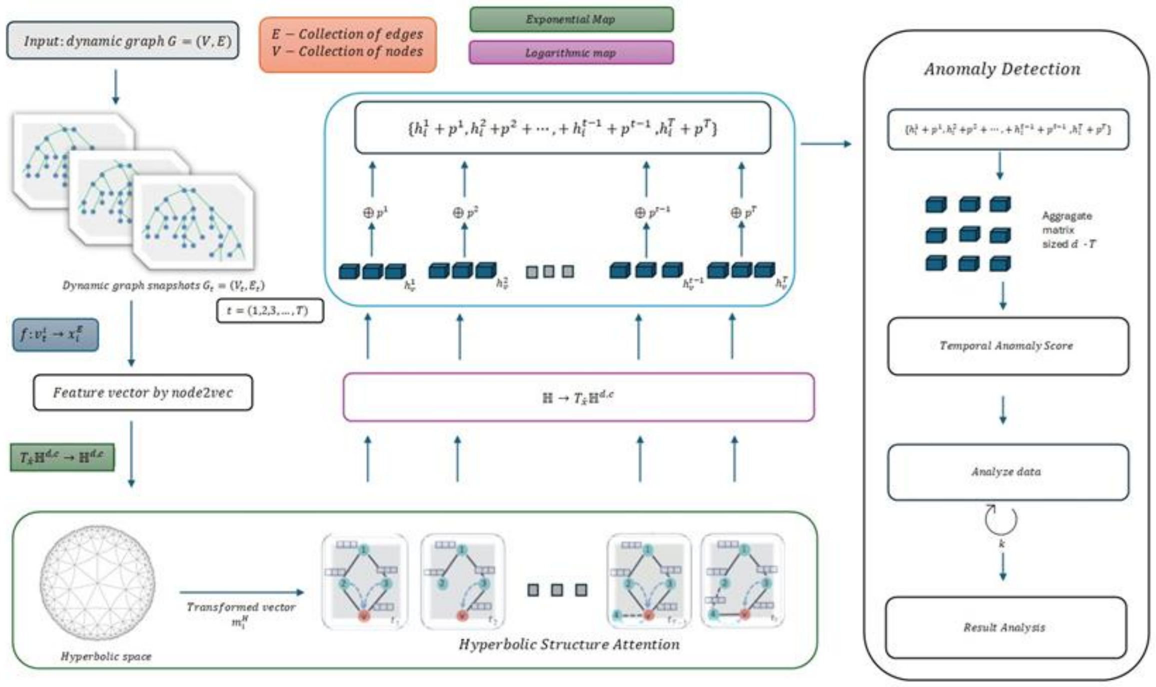
3.4 Sleeping Beauty-

A “Sleeping Beauty” [8] refers to a node in a dynamic network that remains unnoticed or insignificant for an extended period but suddenly becomes highly influential.
This phenomenon often occurs when new discoveries or changing circumstances render the node relevant.
For example, an old research paper might attract significant attention years later if it becomes pertinent to a newly emerging field.

3.5 Falling Star-

A “Falling Star” [4] refers to a node in a dynamic network that begins as highly influential but gradually loses its prominence over time.
This decline can occur due to network evolution, the emergence of more important nodes, or a reduction in the activity or relevance of the original node. For instance, in citation networks, a paper may initially receive significant attention but eventually be overshadowed by newer research.

4. Architecture of the Citation Anomaly Detection Framework



Input: dynamic graph $G = (V, E)$
E – Collection of edges
V – Collection of nodes
Exponential Map
Logarithmic map
Anomaly Detection
$\{h_i^1 + p^1, h_i^2 + p^2 + \cdots, + h_i^{t-1} + p^{t-1}, h_i^T + p^T\}$
$\oplus p^1$
$\oplus p^2$
$\oplus p^{t-1}$
$\oplus p^T$
h_v^1
h_v^2
h_v^{t-1}
h_v^T
Aggregate matrix sized $d \cdot T$
Dynamic graph snapshots $G_t = (V_t, E_t)$
$t = (1,2,3,\ldots,T)$
$f: v_t^i \to x_i^E$
Feature vector by node2vec
$\mathbb{H} \to T_{\bar{x}}\mathbb{H}^{d,c}$
Temporal Anomaly Score
$T_{\bar{x}}\mathbb{H}^{d,c} \to \mathbb{H}^{d,c}$
Analyze data
k
Transformed vector m_i^H
Hyperbolic space
Hyperbolic Structure Attention
Result Analysis

Fig.2 Project Flowchart

4.0 Preparing the Data

We prepared our dataset from the DBLP version 13. DBLP V13 consists of 5,354,309 papers and 48,227,950 citations. Because this dataset is very big, we reduced him to a small dataset for the training. In order to reduce him we first take 3 main clusters, computer science, Nonlinear, and Artificial intelligence. Then we reduce the paper by checking if the abstract fields are not empty and checking if the paper consists of reference files (means the papers are cited).

Our dataset consists of 11,409 papers (nodes) and 10,780 citations (edges) in a range of 61 years from 1959 to 2020. We constructed our dataset based on version 13 of the DBLP database, which includes the following fields in a csv format:-

id	title	authors.name	year	fos.name	n_citation	references	abstract

To process the data, we used a Bag-of-Words (BoW) approach to generate a feature map. Then, we constructed a snapshot graph to serve as an input to the Nod2Vec algorithm.

4.1 Hyperbolic Structural Attention (HSA)

The input to this module is a dynamic graph represented as a sequence of snapshots $G = \{G_1, G_2, \ldots, G_T\}$. It consists of the following steps:

4.1.1 Feature Vector Construction

To construct a feature vectors from the dynamic graph $G = \{G_1, G_2, \ldots, G_T\}$, we applied the Node2Vec algorithm [7] to each snapshot G_t, treating it as a static graph. For each G_t, the input to Node2Vec includes:

- the graph structure (nodes and edges)
- the number of random walks per node-

  We generated **200 walks per node**, which is relatively high and was deliberately chosen to ensure:

  - **High sampling coverage** of each node's local and extended neighborhood.
  - **Stable and reproducible embeddings**, as the skip-gram model has more sequences to learn co-occurrence statistics from.

This setting is particularly important for sparse or evolving graphs, where some nodes appear in only a few snapshots or have low connectivity.

- the walk length
  Each random walk consists of 30 steps. This length is sufficient to traverse both local and moderately global structures within citation graphs, capturing indirect citation chains without drifting too far and introducing noise.

- workers-
  Each random walk consists of 30 steps. This length is sufficient to traverse both local and moderately global structures within citation graphs, capturing indirect citation chains without drifting too far and introducing noise.

- and the hyperparameters p (return parameter) and q (in-out parameter) which controls the bias of the random walks.

Node2Vec generates an embedding for each node, resulting in a feature vector for the snapshot G_t.

To capture the temporal dynamics of the graph, these feature vectors can be aggregated across snapshots using techniques such as concatenation, averaging, or a recurrent neural network (RNN).

This methodology effectively integrates both structural and temporal information from the dynamic graph into the resulting node representations.

4.1.2 Embedding Feature Vectors in Hyperbolic Space

To embed a Euclidean feature vector $x_i^E \in R^d$ into hyperbolic space, we apply the exponential map, assuming the reference point $x' = 0$.
centered at the origin 0.
Under this assumption, the exponential map to the hyperbolic space (e.g., the Poincaré ball) is defined as:

$$\boxed{(7)\ \ exp_0^c(a) = \ (tanh(\sqrt{c}\,||a||)\frac{a}{\sqrt{c}\,||a||})}$$

Here, $c > 0$ is the curvature of the hyperbolic space, and $||\cdot||$denotes the Euclidean norm.

Thus, each Euclidean feature vector $x_i^E \in R^d$ is projected to its hyperbolic counterpart via exponential map $x_i^H = exp_0^c(x_i^E)$.

This transformation is essential for preserving the geometric structure of the original feature space while enabling downstream processing in hyperbolic space, which is particularly effective for capturing hierarchical and scale-free patterns in graph data.

By embedding the vectors in hyperbolic space, we can better represent their relative positions and relationships, reflecting each node’s role and structural importance within the network.

4.1.3 Transformation into Higher Representation

The vector x_i^H undergoes a transformation that enhances its structural expressiveness within the hyperbolic space. This transformation is defined as:

$$\boxed{(8)\ m_i^H = W \otimes^c x_i^H \oplus^c b}$$

Where:

- W is the weighted matrix,
- $\otimes^c$ represents Möbius matrix-vector multiplication:

$$\boxed{(9)\ W \otimes^c x_i^H = exp_{x'}^c(W\,log_{x'}^c(x_i^H))}$$

- $\oplus^c$ denotes mobius addition
- and b is a bias vector in the tangent space, providing flexibility to the model.

This transformation consists of two main operations:

a. **Multiplication in Hyperbolic Space:**
   The operation $W \otimes^c x_i^H$ is carried out using an exponential map in hyperbolic space as seen in eq. 9

b. **Mobius Addition:**
   The second operation, Mobius addition, is given by: $(W \otimes^c x_i^H) \oplus^c b$

The multiplication $W \otimes^c x_i^H$ (eq. 9) in hyperbolic space consists of three steps:

1. **Projection into the Tangent Space**:
   First, the hyperbolic vector x_i^H is projected into the tangent space $T_0 H^{d,c}$ at the origin via the logarithmic map $log_0^c(x_i^H)$ (Fig.3) where:

$$\boxed{(10)\ \ log_0^c(x_i^H) = \frac{1}{\sqrt{c}} \cdot arctanh\left(\sqrt{c} \cdot \left\|x_i^H\right\|\right) \cdot \frac{|x_i^H|}{\left\|x_i^H\right\|}}$$

   This projection linearizes the hyperbolic space locally, enabling the application of standard linear operations. .

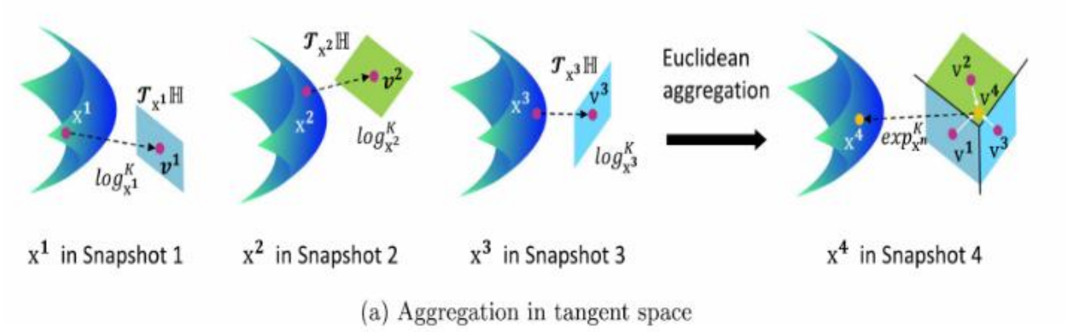


Fig.3

2. **Linear Transformation**:
   Next, a standard linear transformation [11] is applied in the tangent space by multiplying the weight matrix W with $log_0^c(x_i^H)$:

$$\boxed{h_i = W \cdot log_0^c(x_i^H)}$$

   This step enriches the feature representation by capturing structural interactions among nodes.

3. **Projection Back into Hyperbolic Space**:

   Finally, the transformed vector is mapped back to the hyperbolic manifold (Fig.4) using the exponential map exp_0^c:

$$\boxed{m_i^H = exp_0^c(h_i) \oplus^c b\ .}$$

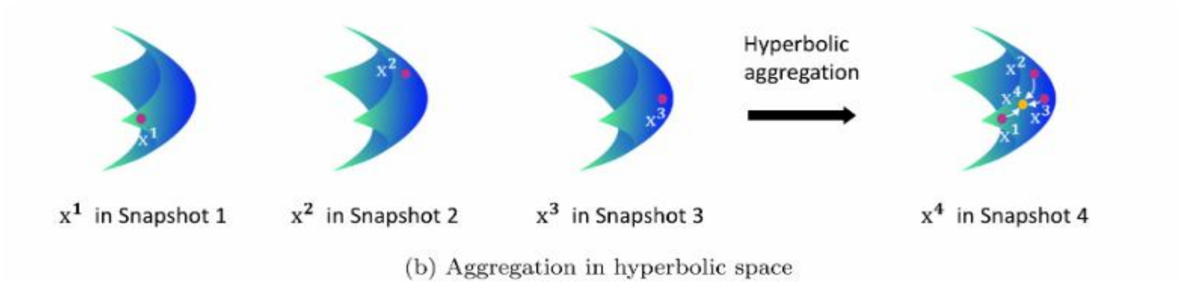


Fig.4

The Mobius Addition $exp_0^c(h_i) \oplus^c b$ with bias vector b yields a non-linear transformation that preserves the hyperbolic geometry and incorporates both learned weights and bias. It ensures flexibility and expressiveness, while respecting the underlying hyperbolic structure. This operation is essential for maintaining geometric consistency during network training.

4.1.4 Hyperbolic Self-Attention Mechanism

To model relationships within the dynamic citation network, we apply a hyperbolic self-attention mechanism. This mechanism serves several key purposes:

1. Modeling Citation Relationships: Self-attention enables each node to attend to its neighbors dynamically weighting their influence based on structural similarity and importance within the graph hierarchy.
   Incorporating Semantic Information: By integrating textual features (e.g., paper abstracts), the attention mechanism helps identify semantic inconsistencies or anomalies, such as unexpected references or thematic mismatches.

2. Enhancing Hyperbolic Representations: Attention selectively emphasizes structurally and semantically relevant nodes, thereby improving the expressiveness of hyperbolic embeddings, which are particularly suited for representing hierarchical and scale-free structures.

This integrated approach ensures that both the structural and semantic properties of the citation network are effectively modeled, facilitating robust downstream tasks such as anomaly detection.

4.1.4.1 Attention Score

Given two hyperbolic-transformed vectors m_i^H ,m_j^H, we compute their relationship via a weighing learned attention mechanism.

First, the vectors are mapped to the tangent space at the origin using the logarithmic map: $$(11)\ s_{ij} = \sigma(a^T([log_0^c(m_i^H m_i)||log_0^c(m_j^H\ m_j)]))$$

Where:

- σ is the sigmoid function, ensuring the score lies in the range $[0,1]$,
- a is a learnable weight vector,
- $||$ denotes vector concatenation.

To normalize these attention scores over the neighbors of node i, we apply the softmax function:

$$\boxed{(12)\ a_{ij} = softmax_{j \in N(i)}(s_{ij}) = \frac{exp(s_{ij})}{\Sigma_{j \in N(i)} exp(s_{ij})}}$$

where $N(i)$ denotes set of neighboring nodes of node i. This normalization ensures that attention weights a_{ij} are comparable and sum to 1, allowing the model to emphasize the most relevant neighbors while maintaining global consistency within the network.

4.1.5 Data Aggregation

To aggregate information from neighboring nodes while respecting hyperbolic geometry, the following steps are performed:

1. The hyperbolic transformed vector m_j^H is first projected into the tangent space using the log_0^c function.
2. The resulting vectors are weighted by their corresponding attention scores a_{ij}.
3. The weighted vectors are summed in the tangent space to form an intermediate representation.
4. This aggregated result is then transformed back into hyperbolic space using the exp_0^c function: $h_i^H = exp_0^c\left(\Sigma_{j \in N(i)} a_{ij} \cdot log_0^c(m_j^H)\right)$.

This procedure allows the model to incorporate both local and hierarchical information in a geometry-aware manner.

4.1.6 Transition to Euclidean Space

For compatibility with downstream modules and to simplify computation, the aggregated hyperbolic representation is mapped back to Euclidean space using the logarithmic map:

$$\boxed{h_{i,t}^E = log_0^c(m_j^H)}$$

This transition retains structural information while enabling further processing in standard Euclidean-based models.

The resulting Euclidean embeddings serve as input to the Euclidean Temporal Attention (ETA) module, which captures temporal dependencies and dynamic patterns in the evolving citation network

4.2 Position Embedding

While node embeddings capture structural and semantic information, they lack awareness of temporal ordering. To address this, we incorporate positional encodings that reflect each node's position within the timeline. This technique enables the model to distinguish between different positions within a sequence.

Given the Euclidean representation $h_{i,t}^{E}$ at time step i , we add a learnable positional position embedding p_t form the temporally enriched representation:

$$\boxed{\{h_i^1 + p_1, h_i^2 + p_2, \ldots, h_i^T + p_T\}} \ .$$

This embedding provides the model with a notion of sequential progression, enabling it to differentiate between node states across time. The result is a set of temporal representations that combine structural information in Euclidean space with the dynamics of the evolving graph.

4.3 Anomaly Detection via Isolation Forest (IF)

Isolation Forest is employed to detect anomalous behavior in the dynamic citation network based on the node representations generated by the previous stages of the model. This section outlines the input preparation, model training, and analysis of anomaly detection results.

4.3.1 Preparing the Input

Given the position-enhanced representations

$$\boxed{\{h_i^1 + p_1, h_i^2 + p_2, \ldots, h_i^T + p_T\}} \ ,$$

the data is organized into a 3D tensor of shape $|v| \times d \times T$, where $|v|$ is the number of nodes in the graph, d is the embedding dimension, and T is the number of time steps.

The tensor is then flattened along the temporal axis to produce a comprehensive vector h_i^E per node i in the size of $d \times T$:

$$\boxed{h_i^E = [h_i^1 + p_1 ||, h_i^2 + p_2 ||, \ldots, || h_i^T + p_T]}$$

To enhance robustness and ensure feature comparability, a normalization technique such as Min-Max Scaling is applied, to rescale features into a standard range (typically [0, 1]) thus making the data suitable for anomaly detection tasks such as Isolation Forests.

4.3.2 Model Training and Anomaly score

Selecting an appropriate model training and anomaly detection framework depends on the dataset's characteristics, including its size, dimensionality, and computational constraints. This selection is critical to ensure both efficiency and accuracy in the analysis.

The anomaly detection is conducted using **Isolation Forest**, a tree-based ensemble method that isolates outliers through random partitioning. The selection of Isolation Forest is motivated by its efficiency, scalability, and suitability for high-dimensional data.

We utilize the implementation provided by Scikit-Learn, a versatile, user-friendly widely-used Python library offering tools for machine learning and data analysis. Scikit-learn supports a variety of algorithms (e.g., One-Class SVM, DBSCAN), but Isolation Forest is particularly well-suited for unsupervised anomaly detection in sparse or irregular data, such as dynamic citation networks.

After training the model on the normalized input vectors h_i^E , each node receives an anomaly score, where higher scores indicate a greater likelihood of anomalous behavior.

4.3.3 Result Analysis

The analysis of anomaly scores involves the following components:

1. Representation of Anomaly Scores

To facilitate the analysis of the anomaly scores, anomaly scores are stored in a **Pandas DataFrame**, with each row representing a single observation and including:

- Node ID: Unique identifier assigned to the node (e.g., AS number).
- Timestamp: The precise time at which the anomaly score was recorded.
- Anomaly Score: The corresponding anomaly score for the node at the given timestamp.

This structured approach enables efficient data manipulation, filtering, and visualization, providing a robust foundation for comprehensive analysis.

2. Basic Statistical Analysis

To identify patterns and potential anomalies, the following statistics are computed per node:

- **Mean Anomaly Score:** Average core per node over all time steps.
- **Variance**: Measures the fluctuation of scores across time.
- **Network Comparison**: Each node's mean anomaly score is compared against the global network average to flag persistent outliers.

Nodes that consistently deviate from the network average or demonstrate abnormally high variance will be considered as potential candidates for further in-depth analysis.

3. Normalization

To ensure consistency and comparability across varying scales of anomaly scores, a normalization process is applied. Specifically, min-max normalization is used to rescale scores to a standardized range of 0 to 1.

This normalization is essential for enabling meaningful comparisons between nodes and facilitating clear and effective visualizations.

4. Visualization of Anomaly Scores

To derive insight into the temporal behavior of nodes, several visualization techniques is employed:

- **Scatter Plots**: Plotting anomaly scores over time for selected nodes or selected node groups, enabling a detailed examination of specific temporal trends, and revealing localized spikes or persistent trends.
- **Line Graphs**: Displaying average anomaly scores per time step for the entire network, as well as for specific node groups to highlight deviations and emerging trends.

These visualizations facilitate an intuitive and systematic identification of nodes exhibiting anomalous behavior and significant temporal trends.

5. Temporal Change Analysis

To detect significant changes over time:

- Snapshot-to-Snapshot Differences:
    - The difference in anomaly score for each node between consecutive time steps is calculated.
    - Nodes with significant changes are flagged as potentially event-driven anomalies, possibly indicating interventions or abrupt shifts in citation patterns.

6. Identification of Extreme Cases

Two types of extreme nodes are identified:

- High Anomaly Scores: Nodes that consistently exhibit scores significantly above the network average.
- High Temporal Volatility: Nodes with the largest score fluctuations across time.

These cases are prioritized for further investigation, potentially indicating structural, topical, or semantic irregularities within the citation network.

5. Research Flow

The research process was structured into two main phases:

5.1 Phase A: Theoretical and Methodological Foundations

This phase established the theoretical, conceptual, and methodological groundwork necessary for designing an effective framework for anomaly detection in dynamic citation networks, leveraging hyperbolic geometry and self-attention mechanisms.

5.1.1 Literature Review and Conceptual Understanding

The first step involved a comprehensive literature review focusing on: dynamic graph representation learning, anomaly detection in temporal networks, hyperbolic embedding techniques, temporal modeling using positional encodings, and Real-world citation anomalies (e.g., *Sleeping Beauty*, *Falling Star*).

A central reference was the paper **"***Dynamic Network Embedding in Hyperbolic Space via Self-Attention***"** (Duanet al., 2022), which introduces a framework for modeling evolving graphs using self-attention in hyperbolic space. This work served as a foundational basis for our approach, particularly for hierarchical and scale-free properties of citation networks.

We also examined various embedding methods such as Node2Vec, DeepWalk, GCN, and GAT, alongside anomaly detection methods including Isolation Forest, and explored the role of the Poincaré ball model in representing hierarchical structures effectively.

5.1.2 Methodological Framework Design

Based on insights from the literature, we designed a custom architecture inspired by the DynHAT model, consisting of three core components:

1. **Hyperbolic Structural Attention (HSA)**: Embeds node features into hyperbolic space and applies self-attention to capture meaningful structural dependencies.
2. **Position Embedding Module**: Encodes temporal information to preserve sequential order and context of citation behavior.
3. **Anomaly Detection**: Applies Isolation Forest to the flattened hyperbolic-temporal embeddings to identify irregular citation patterns.

The use of hyperbolic space stemmed from its superior capacity to model hierarchical and tree-like relationships, which are characteristic of citation networks, unlike Euclidean space, which is limited in capturing such structure

5.1.3 Validation Strategy and Experimental Design

To assess the proposed framework, we designed a validation protocol comprising:

- **Noise injection:** Introducing artificial perturbations (random nodes and edges) into the dynamic graph.
- **Robustness evaluation:** Measuring sensitivity via changes in anomaly scores and classification accuracy.
- **Behavioral comparison:** Comparing outputs between original and perturbed versions of the network.

Key experimental decisions included:

- **Time slicing**: Grouping citation data into annual snapshots
- **Embedding dimensionality**: Balancing expressiveness with computational cost.
- **Noise parameters**: Tuning the number of injected nodes, edge probabilities, and iteration counts based on observed performance.

5.2. Phase B: Implementation and Data Processing

In this phase, we transitioned from design to implementation, using Python 3.10 in the VSCode development environment. This setting provided flexibility in managing virtual environments and Python dependencies.

5.2.1 Model Adaptation

Key features were extracted from the DynHAT [2] module and adapted to fit our architecture. This involved: Parameter tuning, Integration of additional libraries, Structural adjustments to fit the customized model defined in Phase A.

5.2.2 Dataset Construction

Finding a suitable dataset was a significant challenge. We required a citation network filtered by publication year and enriched with metadata such as paper id, title, author and subject classification. However, no publicly available dataset fully met these requirements.

To resolve this, we created a custom dataset from the DBLPv13 dataset (25.6Gb). After filtering and preprocessing, we constructed a tailored dataset of 12 MB, using Bag-of-Words (BoW) to generate a feature map capturing textual and topical similarities. This feature map was then embedded using Node2Vec, producing Euclidean embeddings for each node.

After multiple iterations of experimentation and parameter tuning, we determined the optimal time slicing strategy for our dataset and aligned it with the model's structure.

5.2.3 Model Validation

To validate the model's performance, we generated visual plots and quantitative metrics to analyze its outputs. Once the results aligned with our expectations, we conducted final validation using the noise injection protocol described in Phase A, confirming the model's robustness and accuracy in anomaly detection.

5.2.3.1 Parameter Selection Rationale for Noise Injection Validation

To rigorously test the robustness of our anomaly detection model, we implemented a **noise injection validation protocol**, in which synthetic nodes were inserted into the citation graph to simulate abnormal patterns. Each parameter in the validation procedure was carefully chosen to reflect realistic stress-testing conditions, as detailed below:

- **`n_iters=30`**:
  We repeated the noise injection process over 30 independent iterations to ensure statistical reliability of the results. A high number of runs allows us to mitigate the effect of random fluctuations and obtain stable estimates of model robustness under synthetic perturbations. Repeating this process more than 30 times (e.g., 50 or 100) significantly increases computational cost and GPU

memory usage with diminishing returns. In practice, results tend to converge well before 30 iterations in stable models. Using a value lower than 30 (e.g., 5 or 10), increases the variance in results due to randomness in graph structure and noise features.Reduces the confidence in anomaly detection behavior.

- **`n_noise_nodes=10`**:
  Each iteration injected 10 synthetic nodes into the network. This number strikes a balance between introducing sufficient noise to challenge the model, without overwhelming the graph or skewing the overall node distribution. It simulates the presence of a small fraction of potentially anomalous papers entering the network.

- **`frac_above_95` metric**:
  To quantify the anomaly magnitude of injected noise, we computed the fraction of noise nodes whose anomaly scores exceeded the 95th percentile of original nodes. A high fraction here confirms that the model reliably assigns high anomaly scores to synthetic outliers, validating its power.

- **Validation Accuracy (`val_acc_noisy`)**:
  By evaluating prediction accuracy on the validation set after noise injection, we assessed how well the model generalizes in the presence of structural noise augmentations. Stable accuracy across iterations implies robustness to noise without compromising core prediction capabilities.

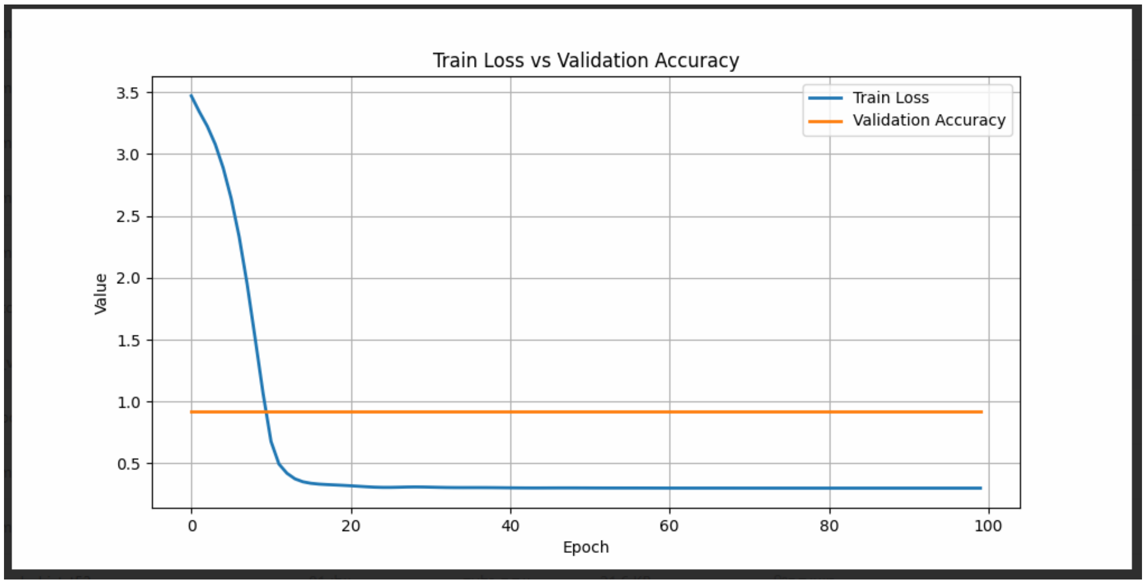


Fig.5 Monitored Model Performance Graph

Using the standard validation methods, we performed a visualization(see Fig.5) of the train loss and validation accuracy across the epochs. As a result of that, we can see that the train loss(blue line) drops sharply in the first 10-15 epochs, indicating the model learning to fit the training data very well,while the validation accuracy(orange line) remains constant throughout all epochs, at about ~0.918%,indicating high accuracy in the model process.

6. Tools & Softwares used

The implementation of the project relied on a diverse set of tools and libraries spanning scientific computing, machine learning, graph processing, and deep learning frameworks for graph neural networks. Below is a categorized overview of the main components used:

1. Scientific Computing

Library	Description
`numpy`	Used for efficient numerical computation, including vectorized operations and reshaping.
`pandas`	Handled tabular metadata such as paper ID, title, publication year, and citation references.
`matplotlib`	Provided visualization such as anomaly score traces, histograms, and scatter plots.

2. Machine Learning

Library	Description
`scikit-learn` (`sklearn.ensemble.IsolationForest`)	Applied for unsupervised anomaly detection via isolation trees.

`sklearn.feature_extraction.text.CountVectorizer`	Converted textual metadata to a sparse matrix of token counts, facilitating the integration of text-based node features.
`sklearn.preprocessing.LabelBinarizer`	Encoded categorical variables into binary arrays, supporting potential multi-label classification tasks.

3. **Graph Processing**

Library	**Description**
`networkX`	Used to construct and manipulate dynamic citation graphs structure.
`scipy.sparse`	Provided sparse matrix representations of adjacency matrices to optimize memory usage and computational efficiency.

4. **Graph Neural Networks (PyTorch & PyG)**

Library	**Description**
`torch`	Core deep learning framework for tensor operations, model training, and GPU acceleration.
`torch.nn.functional`	Provided differentiable operations, including loss as cross-entropy).
`torch_geometric` (PyG)	Enabled conversion of graph data from NetworkX and SciPy graphs formats to GNN-compatible `edge_index` format using utilities like `from_networkx and from_scipy_sparse_matrix`.

5. Node Embeddings

Tool	Description
`Node2Vec`	Learned structural node embeddings using biased random walks combined with the skip-gram model. Used for initial feature extraction before projection to hyperbolic space.

7. Results and Discussion

The main goal of this project was to detect anomalies ןn dynamic citation networks, with a specific focus on identifying two behavioral archetypes: “Sleeping Beauty” and “Falling Star”. These represent temporally unusual citation trajectories, where a paper either gains delayed impact (Sleeping Beauty) or experiences a brief spike in attention (Falling Star) followed by decline.

7.1 Temporal Anomaly Patterns: Sleeping Beauty

In the following graph(Fig.6) we present the anomaly score trajectories of the top five nodes exhibiting the most significant fluctuations over time. The Y-axis represents the anomaly score, while the X-axis represents the time step. Red dots highlight sharp changes.
Most nodes display a stable and low anomaly score over time, corresponding to a “sleeping phase” characterized by minimal activity. Suddenly, a subset of nodes shows a **sharp spike** in their anomaly scores, indicating an abrupt structural or semantic change, likely due to a burst of citations. This constitutes the “Awaking phase”, which indicates a notable event in the citation network, meaning the nodes were cited. After the peak, the score gradually declines in a “post-awakening phase”, signaling a return to typical behavior.
This pattern provides a textbook example of the Sleeping Beauty phenomenon in citation network dynamics: a paper that initially receives little attention, suddenly gains prominence, and then stabilizes.

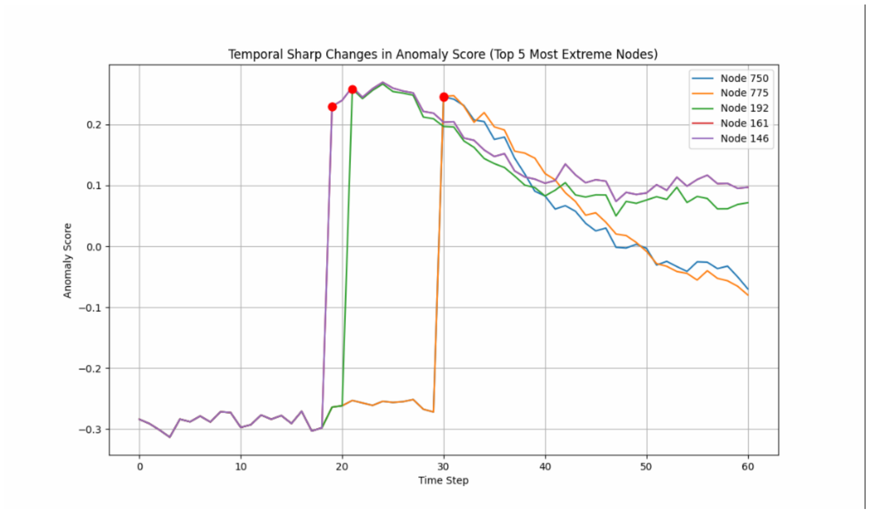


Fig. 6: Temporal Anomaly Score Spikes Indicating Sleeping Beauty Emergence

7.2. Falling Star Pattern Analysis via ∆ Anomaly Score (ΔAS)

To capture transient citation anomalies, we analyze the standard deviation of the change in Anomaly Score (ΔAS) across several time windows. Specifically, we identify nodes with high standard deviation in ΔAS, which reflects abrupt temporal changes in anomaly behavior. The analysis spans three distinct intervals: [10–20], [15–25], and [40–50]. For each window, we identified the top 10 nodes with the highest ΔAS standard deviation, highlighting temporal instability.

Time Window: [10–20]:

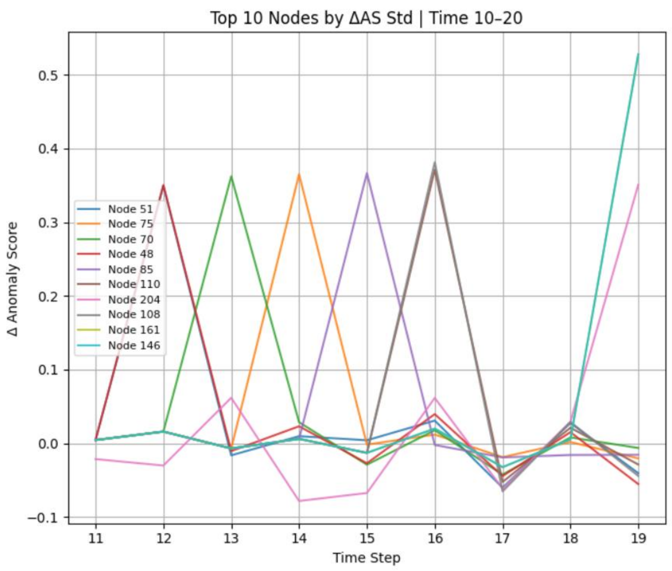


Fig.7: Detection of Falling Star Patterns via ΔAnomaly Score Fluctuations (Time 10–20)

In this interval (Fig 7), Several nodes exhibit short-lived bursts in anomalous behavior:

- Node 75 peaks sharply at time step *t = 13* (with a ΔAS of ~0.35), followed by a rapid decline.
- Nodes 70 and 85 display similar patterns, at steps *t = 12* and *t = 14* respectively.

These temporal patterns align with the Falling Star phenomena: a node gains brief attention, possibly due to a sudden increase in citations, attention, or network activity,

Time window: [15-25]:

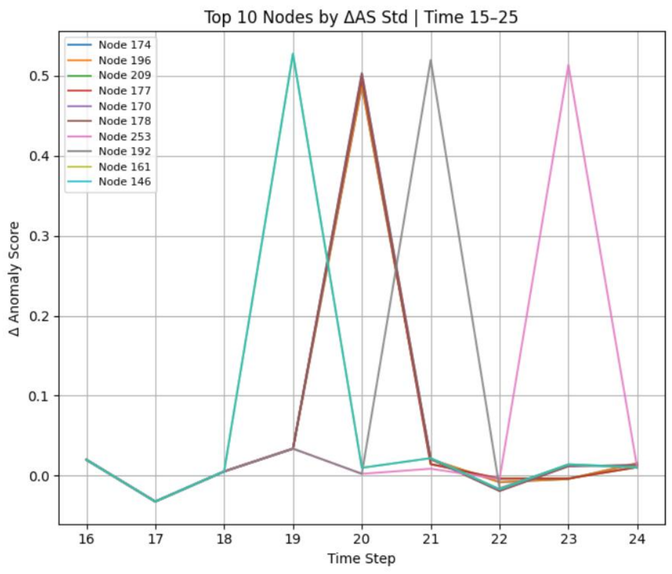


Fig.8: Detection of Falling Star Patterns via ΔAnomaly Score Fluctuations (Time 15–25)

The most prominent and clearest Falling Star patterns are observed in this interval (Fig.8):

- Node 209 reaches a ΔAS maximum (~0.55) at step *t = 20*, followed by an immediate drop
- .
  Nodes 170 and192 exhibit analogous patterns at *t = 21* and *t = 23*, respectively.

Time Window: [40-50]:

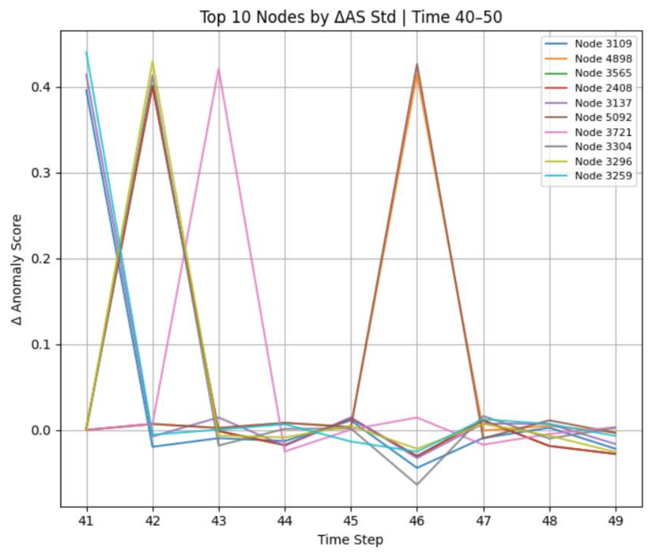


Fig.9: Detection of Falling Star Patterns via ΔAnomaly Score Fluctuations (Time 40–50)

Later in the timeline, nodes 4898, 3565, and 3259 demonstrate clear ΔAS peaks at steps *t = 42*, *t = 46*, and *t = 41*, respectively as shown on fig.9. e Despite occurring in a later timeframe, the pattern remains consistent with the Falling Star signature.

Unlike earlier intervals, these peaks appear to be slightly less synchronized across nodes, potentially indicating isolated or topic-specific citation shifts rather than global network changes.

This analysis reinforces the effectiveness of ΔAS standard deviation as a proxy for capturing non-stationary anomalies, especially in evolving scientific literature or citation networks. Furthermore, such transient anomalies may correlate with media exposure, rapid community attention, or event-driven citations. Integrating metadata such as publication venue, authorship centrality, or social attention could further illuminate the drivers behind these spikes.

7.3 Mean Anomaly Score Distribution: Evidence of Dual Behavioral Regimes

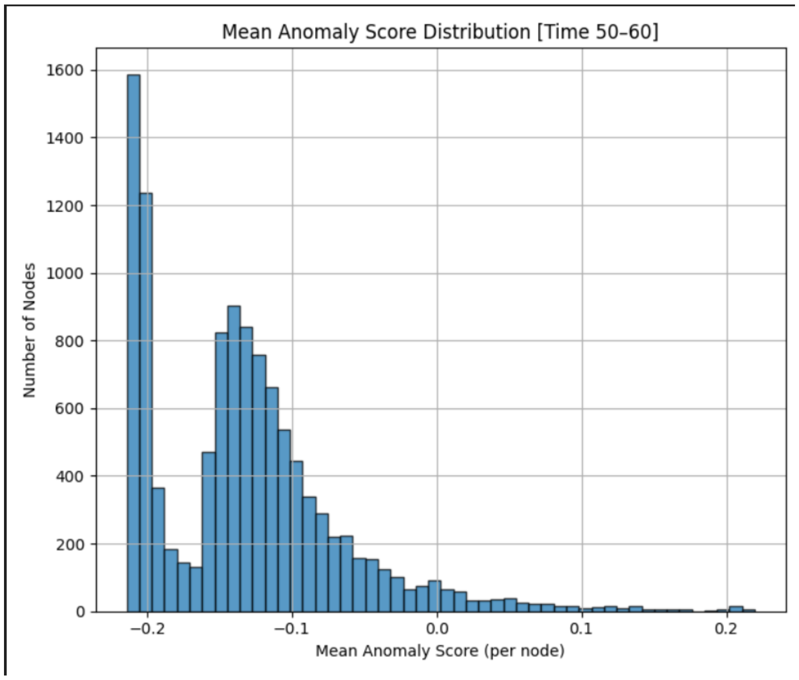


Fig.10: Histogram of Mean Anomaly Scores Highlights Non-Zipfian Patterns in Citation Dynamics

Fig.10 presents a histogram of mean anomaly scores per node across time steps 50 to 60. A key observation is the bimodal nature of the distribution, characterized by two distinct peaks:

❖ A primary peak centered around -0.2, representing the majority of nodes with consistently low negative anomaly scores,likely papers that received few or no citations.
  A secondary cluster between -0.12 and -0.1 forming a plateau and revealing a substantial population with moderate irregular citation behavior.
❖ A long right tail, though sparse, extends toward positive anomaly scores (up to +0.2), indicating the presence of rare but detectable high-anomaly nodes (e.g., Sleeping Beauties or Falling Stars).

7.4 Interpretation Beyond Zipfian Behavior

While many real-world networks follow Zipfian (power-law) distributions, with few highly active nodes and many low-activity ones, the presence of two distinct peaks challenges the assumption of a unimodal, heavy-tailed distribution.

Rather than a unimodal, heavy-tailed Zipfian curve, this histogram indicates that the model reveals two behavioral subpopulations:

1. **Baseline Nodes**: Regular citation patterns, forming a sharp low-score peak.
   - The sharp peak at approximately -0.2 corresponds to a large cluster of nodes with consistently low or no anomalous behavior.
     These nodes likely exhibit regular temporal citation dynamics and align with structural normality in the graph.

2. **Mildly Anomalous Nodes**: Subtle deviations, possibly indicating early-stage citation surges or thematic transitions.
   - The secondary rise (centered around -0.1 to -0.12) suggests a non-negligible group of nodes deviating from the dominant trend.
     These may represent latent anomalies or entities undergoing subtle behavioral changes over time, e.g., early-stage citation surges or transitions in topical influence.

This multimodal structure suggests that the model captures latent dynamics beyond what Zipfian or static centrality measures typically expose. The anomaly-aware representation detects both sustained and transient irregularities, offering a richer view of citation evolution.

7.5 Broader Implications

This multimodal structure suggests that the model captures more than structural noise:

- It distinguishes qualitatively different behavioral regimes among nodes.

- It identifies subtle group-level deviations that a Zipfian model or static centrality measure might obscure.

- The clustering in the distribution potentially reflects emerging dynamics, such as thematic shifts, author collaborations, or community realignments in the citation network.

The findings open several avenues for future research:

- **Dynamic ranking models:** Incorporating temporal anomaly patterns as a supplement to classic impact metrics (e.g., Impact Factor).
- **Early warning signals:** Identifying emerging trends or influential papers through anomaly score spikes.
- **Metadata integration:** Correlating anomaly events with publication venue, topic, or author characteristics to understand citation behavior drivers

- In summary, this project demonstrates that temporal anomaly detection in hyperbolic space provides a nuanced lens for understanding scientific influence, beyond static or aggregated citation measures. The model effectively captures both Sleeping Beauty and Falling Star patterns, highlighting the value of integrating structural, semantic, and temporal dimensions in scholarly network analysis

8. Challenges We Faced

Throughout the development of the project, we encountered several significant challenges:

1. **Dataset Availability and Construction**

The primary challenge was finding a dataset that met our requirements. Initially, we attempted to use the Cora dataset, but it contained missing values and lacked critical metadata. After an extensive search, we concluded that no existing dataset fully satisfied our needs. In consultation with our supervisor, we identified a metadata source and constructed a custom dataset from it, tailored specifically to our project's goals.

2. **Environment and Library Compatibility**

Another difficulty was selecting the appropriate Python version to ensure compatibility among key libraries, especially node2vec and torch scatter, libraries, which have specific dependency constraints. After multiple trials, we found that Python 3.10 provided the best compatibility and stability for our environment.

3. **Parameter Tuning**

Determining the optimal parameters for our model required significant time and computational effort. The process involved trial and error and led to periods of unproductive waiting, as we observed model behavior and evaluated results iteratively.

4. **Computational Limitations**

The algorithms and libraries used (e.g., GNN layers, attention mechanisms, hyperbolic projections) demand high computational resources, which exceed the capacity of our personal computers. These limitations also caused difficulties in version control and synchronization through Git. Ultimately, we used the GPU resources available at the college for model training and testing.

Key Insights from Project Experience:

- **Early Planning for Computational Resources**

One major lesson was the importance of anticipating computational needs. Initially, we attempted to run the model locally and partially on Google Colab without migrating the full project.
However, due to the heavy memory and GPU demands of our model, and the limited computing resources at Braude College, we concluded that Google Colab Pro was necessary for proper training. In retrospect, the project should have been developed entirely within Colab from the beginning to avoid compatibility and migration issues.

- **Effective Use of GitHub**

GitHub proved to be an essential tool for collaborative development. It enabled us to work asynchronously on different modules and functions while maintaining a centralized and secure version of the codebase. This prevented data loss and ensured code consistency across different development stages.

9. Project Metrics Evaluation

The project successfully met its defined metrics. The core objectives, detecting anomalies in dynamic citation networks using hyperbolic geometry and self-attention mechanisms, were achieved both conceptually and in implementation.

Specifically:

- **Anomaly Detection Accuracy**: The model successfully identified approximately 91.8% of injected anomalies, as measured through controlled noise injection experiments. This aligns with our performance target and validates the model's ability to detect irregular citation behavior.
- **Scalability and Temporal Sensitivity**: The model was able to process temporal snapshots of the citation graph efficiently and to capture shifts in node behavior over time, demonstrating the effectiveness of the temporal attention mechanism in combination with hyperbolic embeddings.
- **Robustness to Noise**: Through repeated noise injection and perturbation experiments, the framework consistently maintained high classification performance, indicating robustness to artificial irregularities introduced in the graph.
- **Computational Feasibility**: Despite some initial limitations, we were able to train and test the model on realistic-scale datasets using available GPU resources, demonstrating practical feasibility.

While some adjustments and workarounds were needed, such as constructing a custom dataset and optimizing parameters, the final results are in line with our project's success criteria.

Bibliography

[1] Chen, J., Xiang, J., Huang, T., Zhao, X., & Liu, Y. (2025, January 7). Hyperbolic binary neural network. arXiv.org. https://arxiv.org/abs/2501.03471

[2] Duan, D., Zha, D., Yang, X., Mu, N., Shen, J. (2022). Dynamic Network Embedding in Hyperbolic Space via Self-attention. In: Di Noia, T., Ko, IY., Schedl, M., Ardito, C. (eds) Web Engineering. ICWE 2022. Lecture Notes in Computer Science, vol 13362. Springer, Cham. https://doi.org/10.1007/978-3-031-09917-5_13

[3] Duan, D., Zha, D., Liu, Z., & Chen, Y. (2024). Dynamic Graph Embedding via Self-Attention in the Lorentz Space. Proceedings of the 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Tianjin, China, pp. 199–204.

[4] Fire, M., & Guestrin, C. (2018, October 13). *The rise and fall of network stars: Analyzing 2.5 million graphs to reveal how high-degree vertices emerge over time*. arXiv.org. https://arxiv.org/abs/1706.06690

[5] GeeksforGeeks. (2024, November 19). *Softmax activation function in neural networks*. https://www.geeksforgeeks.org/the-role-of-softmax-in-neural-networks-detailed-explanation-and-applications/

[6] GeeksforGeeks. (2024, July 15). What is Isolation Forest? GeeksforGeeks. https://www.geeksforgeeks.org/what-is-isolation-forest/

[7] Grover, A., & Leskovec, J. (2016). Node2vec. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 855–864. https://arxiv.org/pdf/1607.00653

[8] Ke, Q., Ferrara, E., Radicchi, F., & Flammini, A. (2015). Defining and identifying Sleeping Beauties in science. Proceedings of the National Academy of Sciences of the United States of America, 112(24), 7426–7431. https://www.jstor.org/stable/26463784

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023, August 2). Attention is all you need. arXiv.org. https://arxiv.org/abs/1706.03762

[10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018, February 4). *Graph attention networks*. arXiv.org. https://arxiv.org/abs/1710.10903

[11] Zhang, R. (2023, December 17). A journey into linear algebra: Exploring linear transformations. Medium. https://rendazhang.medium.com/a-journey-into-linear-algebra-exploring-linear-transformations-5d8b00f36be8