Utilizing Pre-Final Vectors from GNN Graph Classification for Enhanced Climate Analysis

Alex Romanova[0000−0002−5927−2129]

Independent Researcher, McLean, VA, USA
sparkling.dataocean@gmail.com
http://sparklingdataocean.com/

Abstract. This study introduces a novel method for enhancing deep learning models by integrating linear algebra techniques within Graph Neural Network (GNN) Graph Classification models. By extracting pre-final vectors from these models and analyzing the embedded 'small graphs,' we significantly enhance data representation and manipulation. This approach supports advanced tasks, including nearest neighbor identification, clustering, statistical analysis, and meta-graph construction. Although climate time series data is used as an example, this method's versatility extends to various fields—such as identifying key influencers in social networks or detecting anomalies like sudden price spikes in financial markets—thereby uncovering hidden patterns and insights across diverse domains.

Keywords: Graph Neural Networks (GNN) · Graph Classification · Embedded Graphs · Linear Algebra · Climate Data · Time Series · Data Analysis

1 Introduction

Linear algebra is indispensable in the world of machine learning and artificial intelligence, primarily because it provides a way to efficiently represent and manipulate data. Whether dealing with matrices or vectors, these mathematical structures help model complex problems in a computationally manageable form. In recent years, the surge in popularity of deep learning models has highlighted the versatility of linear algebra across various domains [1, 2].

Converting various types of data, such as images, audio, text, and social network information, into a uniform vector format is crucial for deep learning. This standardization facilitates processing and analysis by deep learning algorithms and opens doors to innovative AI applications across multiple domains. The popularity of vector representations, from Word2Vec and Doc2Vec to various x2Vec models, has grown significantly due to their ability to encode complex data in compact numerical forms [3]. This makes tasks like semantic similarity, document classification, and network analysis more efficient . These vectors enable essential machine learning methods, including clustering, classification,

and regression. Linear algebra underpins these processes, allowing the manipulation and analysis of data within neural network pipelines. Each step in these pipelines often involves vector operations, highlighting linear algebra's critical role in advancing deep learning technology.

Transformers that embed text into vectors have revolutionized natural language processing by converting textual data into continuous vector spaces, enabling efficient analysis and manipulation. This technique has significantly improved tasks such as semantic similarity, text classification, and sentiment analysis. The embeddings produced by transformers, such as BERT and GPT, serve as rich, high-dimensional representations of text that can be integrated into other machine learning models [4]. Large Language Models (LLMs) like GPT-4 have built upon transformer architectures to further advance natural language processing. These models leverage the high-dimensional embeddings produced by transformers to understand and generate human-like text. LLMs excel at a variety of language tasks, from answering questions to generating coherent and contextually relevant paragraphs of text.

However, despite their strengths, LLMs have limitations when it comes to handling complex, structured data, such as climate data. LLMs are primarily designed for sequential data and struggle with non-linear relationships inherent in many datasets. They lack the capability to naturally model multi-dimensional relationships. To address these limitations, transformers have been effectively applied in the domain of Graph Neural Networks (GNNs). One application of transformers is GNN Link Prediction on semantic graphs, which involves using transformers for node text feature embedding and GNN Link Prediction for re-embedding based on graph topology. The GraphSAGE Link Prediction model in the Deep Graph Library (DGL) [5] generates pre-final vectors for each node instead of direct link predictions. In our previous studies, we have utilized these vectors to assess node similarities for recommender systems [6] and performed graph triangle analysis to uncover hidden connections in knowledge graphs [7].

This study extends the techniques of capturing pre-final vectors from GNN Link Prediction to GNN Graph Classification models, which remains a largely unexplored area. We aim to demonstrate how these intermediate vectors can be utilized for various graph data analyses beyond their primary classification tasks.

In one of our previous study [8], we used GNN Graph Classification models to classify climate time series data into stable and unstable classes and detect abnormal climate change patterns. In this study, we extend this work by capturing pre-final vectors and embedding small graphs, leveraging the detailed representations provided by these intermediate vectors to enhance climate data analysis. For this study experiments, we explore the same Kaggle dataset [9] and employ the same method of building many small graphs. The raw data consists of daily temperature data spanning 40 years for the 1,000 most populous cities in the world. For each city, we construct a graph where nodes represent city-year combinations. The features of each node are the daily temperature vectors corresponding to that specific city and year. Graph edges are defined by selecting

pairs of vectors with cosine similarities higher than a threshold. In the previous study, for graph labeling, we used climate trends by calculating cosine similarities between daily temperature vectors for consecutive years. In this study, for graph labeling, we classify graphs as 'stable' or 'unstable' based on the city's geographical latitude.

Running the GNN Graph Classification model in that study [8] quickly reached very high accuracy in just a few epochs. Initially, this raised concerns about the results, but further investigation, including experiments with other time series data [10] and semantic graphs [11], revealed that the GNN Graph Classification model's exceptional performance was due to its acute sensitivity to variations in graph topology.

The model's sensitivity to graph topology offers both a risk of overfitting and a powerful advantage in outlier detection. In this study, we aim to enhance the analysis of dynamic datasets by capturing pre-final vectors from GNN Graph Classification models. We will validate extraction techniques and demonstrate their effectiveness in managing and analyzing complex data. By applying similarity measures to these pre-final vectors, we improve tasks such as identifying the closest neighbors for any graph. Furthermore, we construct meta-graphs from these embedded small graphs, advancing the capabilities of graph mining.

2 Related Work

In 2012, a significant breakthrough occurred in the fields of deep learning and knowledge graphs. The introduction of Convolutional Neural Network (CNN) image classification through AlexNet [12] showcased its superiority over previous machine learning techniques in various domains [13]. Concurrently, Google introduced knowledge graphs, enabling machines to understand relationships between entities and revolutionizing data integration and management, enhancing products with intelligent and 'magical' capabilities [14].

The growth of deep learning and knowledge graphs occurred simultaneously for years, with CNNs excelling at grid-structured data tasks but struggling with graph-structured ones. Conversely, graph techniques thrived on graph-structured data but lacked the capability of deep learning. In the late 2010s, Graph Neural Networks (GNNs) emerged, combining deep learning and graph processing, revolutionizing the handling of graph-structured data by enabling complex data analysis and predictions through the effective capture of relationships between graph nodes [15].

Starting in 2022, Large Language Models (LLMs) became prominent in the deep learning landscape, capturing most of the research attention. However, the potential of GNNs continues to be recognized, and we remain hopeful that GNN research and applications will continue to emerge and expand.

GNNs excel in analyzing complex data structures by capturing intricate relationships within graph structures. They have been widely applied in chemistry, medicine, and biology, particularly in GNN Graph Classification, to unravel

complex molecular structures and biomolecules, providing critical insights for therapeutic and treatment advancements [16–18].

A comprehensive survey [19] delves into graph classification and link prediction using Graph Neural Networks, discussing various GNN architectures for classifying graphs and predicting edges. It covers common datasets, benchmarks, and applications in social networks, biology, and recommendation systems, and highlights key challenges like scalability and interpretability. The study also showcases GNNs' proficiency in handling complex non-Euclidean data, such as in social and transportation networks, and their capabilities in spatial-temporal data analysis for urban traffic flow and weather prediction.

The paper [20], introduces a novel method for time series forecasting using Graph Neural Networks. The approach, named TimeGNN, models temporal dependencies in dynamic graphs by conceptualizing time steps as nodes. This allows the model to effectively capture complex temporal patterns and interactions within the data, enhancing scalability and efficiency for large datasets and various window sizes. The authors demonstrate significant improvements in forecasting accuracy across different applications, highlighting the potential of GNNs in analyzing dynamic temporal data.

The application of deep learning, particularly Convolutional Neural Networks (CNNs), in atmospheric imaging and climate data mining has yielded promising results in tasks such as predicting tropical cyclones [21]. However, literature reviews indicate that the use of deep learning in climate data analysis, including weather pattern detection, is still an evolving field [22, 23].

Building on this foundation, our study explores the use of Graph Neural Network Graph Classification models for climate data analysis. Unlike traditional CNN approaches, GNNs allow us to decode complex time series data, providing a novel perspective on understanding and forecasting climate dynamics. This method not only enhances predictive capabilities but also adds a new dimension to environmental science by offering deeper insights into the underlying patterns of climate datasets.

While capturing pre-final vectors from GNN Node Classification and Link Prediction is a well-established area of research, the study of capturing embedded whole graphs from GNN Graph Classification models remains unexplored. This paper presents a novel study that extends the exploration of pre-final vectors to GNN Graph Classification models, aiming to demonstrate how these intermediate vectors can be utilized for various graph data analyses beyond their primary classification tasks. By doing so, we aim to fill the gaps in existing literature and introduce innovative techniques for analyzing and interpreting dynamic datasets.

3 Method

3.1 Graph Construction and Climate Labeling

In this study, we utilized Graph Neural Network (GNN) Graph Classification models to analyze small, labeled graphs constructed from nodes and edges. We

created graphs for each city, where nodes represent specific city-year pairs and edges connect node pairs with cosine similarities above a certain threshold. In our previous study [8], we employed a similar method of building city graphs by tracking climate trends through cosine similarities between daily temperature vectors for consecutive years, allowing us to detect changes in climate patterns. However, in this study, we adopted a simpler approach for graph labeling, classifying graphs as 'stable' or 'unstable' based solely on the city's geographical latitude.

3.2 Implementation of GCNConv for Graph Classification

For classifying the climate graphs, we employed the Graph Convolutional Network (GCNConv) model from the PyTorch Geometric Library (PyG) [24]. This model effectively extracts feature vectors from graph data, enabling binary classification to identify 'stable' or 'unstable' climates.

Beyond classification, we extracted pre-final vectors from the GCNConv model. These vectors provide detailed representations of the small graphs and were further used for additional analyses, such as identifying the closest neighbors and constructing meta-graphs to explore more complex relationships within the climate data.

For more detailed analysis and implementation details, please refer to our technical blog at [25].

4 Experiments

4.1 Data Source

We obtained our primary dataset from Kaggle, titled "Temperature History of 1000 Cities 1980 to 2020" [9]. This dataset provides a comprehensive record of average daily temperatures in Celsius for the 1,000 most populous cities worldwide, covering the period from 1980 to 2019. Using this extensive dataset, we developed a GNN Graph Classification model to analyze and interpret the climatic behaviors of these urban centers.

In our analysis, each city was represented as an individual graph, with nodes corresponding to specific city-year pairs. These nodes encapsulate the temperature data for their respective years, allowing for a detailed examination of temporal climatic patterns within each city.

The graphs were labeled as 'stable' or 'unstable' based on the latitude of the cities. We assumed that cities closer to the equator exhibit less temperature variability and therefore more stability. This assumption aligns with observed climatic trends, where equatorial regions typically experience less seasonal variation than higher latitudes. To categorize the cities, we divided the 1,000 cities into two groups: one consisting of cities nearer the equator and the other comprising cities at higher latitudes.

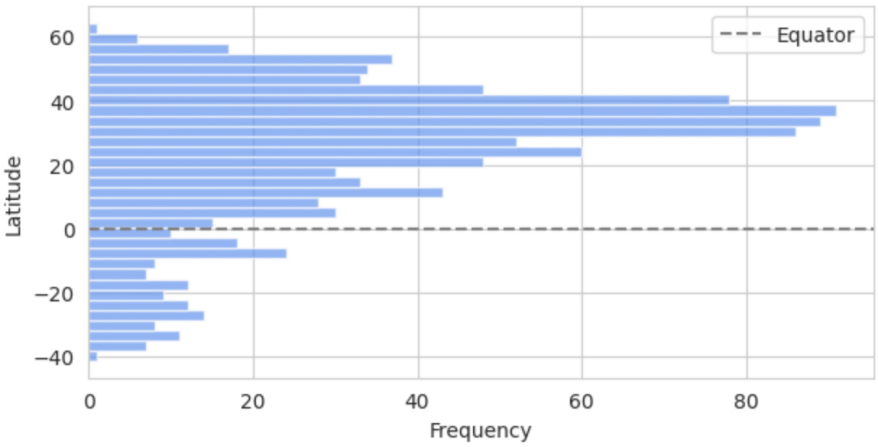

Fig. 1. Global Latitude Distribution of Top 1,000 Most Populous Cities.

Table 1. Closest Neighbors of Tokyo, Japan (Lat 35.69, Long 139.69). Based on Cosine Similarity

City	Country	Latitude	Longitude	Label
Shiraz	Iran	29.63	52.57	0
Beirut	Lebanon	33.87	35.51	1
Luoyang	China	34.68	112.47	1
Latakia	Syria	35.54	35.78	1
Sacramento	United States	38.57	-121.47	1

Table 2. Closest Neighbors of Gothenburg, Sweden (Lat 57.71, Long 12.00). Based on Cosine Similarity

City	Country	Latitude	Longitude	Label
Krakow	Poland	50.06	19.96	1
Katowice	Poland	50.26	19.02	1
Bytom	Poland	50.35	18.91	1
Lodz	Poland	51.77	19.45	1
Gdansk	Poland	54.36	18.64	1

However, it is important to note that determining climate stability is more complex and influenced by various factors beyond latitude, such as regional climate patterns and local environmental conditions. While latitude serves as a general indicator of potential temperature variability, a comprehensive assessment would require a more detailed analysis.

Figure 1 illustrates the latitude distribution of the 1,000 most populous cities worldwide, with the equator marked by a dashed line at 0 degrees latitude. The horizontal bars represent the frequency of cities within each latitude range, showing a higher concentration of cities around mid-latitudes and fewer cities near

the equator and higher latitudes. This distribution highlights the geographic diversity of urban centers considered in our study and provides a foundation for analyzing climatic stability relative to latitude. Understanding this spread is essential for interpreting temperature variability and potential climate stability, as cities closer to the equator are assumed to have less seasonal variation compared to those at higher latitudes.

4.2 Data Preparation and Model Training

In developing the GNN Graph Classification model for climate data analysis, we constructed individual graphs for each city, labeling them as 'stable' or 'unstable' based on latitude. Edges were created between node pairs with high cosine similarities, indicating similar temperature trends. To ensure each graph formed a single connected component, we introduced virtual nodes to enhance connectivity and improve the model's generalization across diverse urban climates.

We utilized the GCNConv model from the PyTorch Geometric (PyG) library [24] for our analysis. This model was used to extract pre-final feature vectors from the graphs, which are crucial for detailed analyses of climate patterns before final classification.

The dataset of 1,000 city graphs was split into 888 for training and 112 for testing. The GCNConv model achieved high accuracy, with approximately 94% on the training data and 92% on the test data, demonstrating its effectiveness in detecting and classifying anomalous climate trends through graph-based representations of daily temperature data.

4.3 Application of Graph Embedded Vectors: Cosine Similarity Analysis

After training the GNN Graph Classification model, we transformed each city graph into an embedded vector. These vectors served as the foundation for subsequent data analyses.

Analysis of Cosine Similarity Matrix of Graph-Embedded Vectors: We constructed a cosine similarity matrix for 1,000 cities to identify closely related climate profiles. This matrix enables nuanced comparisons and clustering based on the embedded vector data.

To illustrate this, we examined the closest neighbors of the graph vectors for Tokyo, Japan (the largest city in our dataset), and Gothenburg, Sweden (the smallest city in our dataset). As shown in Table 1, Tokyo's closest neighbors are predominantly major Asian cities, indicating strong climatic and geographical similarities. Similarly, Table 2 shows that Gothenburg's nearest neighbors are mostly European cities, reflecting similar weather patterns across Northern and Central Europe.

We also identified vector pairs with the lowest cosine similarity, specifically -0.543011, between Ulaanbaatar, Mongolia, and Shaoguan, China. This negative similarity suggests stark climatic differences. Additionally, the pair with

cosine similarity closest to 0.0 (-0.000047), indicating orthogonality, is between Nanchang, China, and N'Djamena, Chad, highlighting the lack of significant relationship between these cities' climatic attributes.

Cosine Similarity Matrix Distribution: The distribution of cosine similarity values from the embedded city graphs shows notable peaks above 0.9 and between -0.4 and -0.2, indicating distinct clustering based on climatic profiles. Some clusters show high similarity, reflecting nearly identical climates, while others display moderate similarities, highlighting shared but less pronounced features.

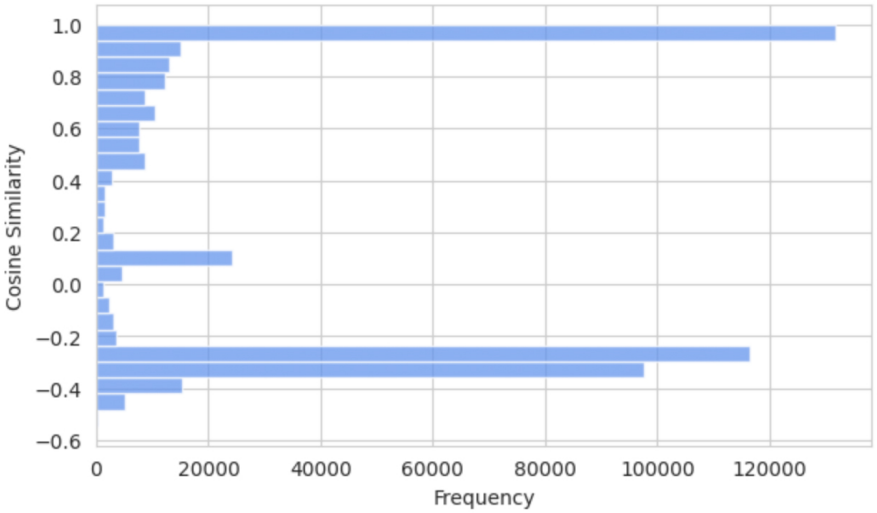

Fig. 2. Cosine Similarity Distribution.

These patterns are graphically represented in Figure 2 and detailed in Table 3. The figure clearly shows the skewed distribution, emphasizing areas with the highest concentration of values. This visualization is crucial for interpreting the relational dynamics between cities, providing insights into the clustering patterns derived from their climate data.

4.4 Application of Graph Embedded Vectors: Graphs Derived from Cosine Similarity Thresholds

Based on the observed distribution of cosine similarities, we generated three distinct graphs for further analysis. Each graph was constructed using different cosine similarity thresholds to examine how these thresholds affect city pair distances.

Table 3. Distribution of Cosine Similarities

Cosine Interval	**Frequency**
-1.0 to -0.6	0
-0.6 to -0.5	59
-0.5 to -0.4	8464
-0.4 to -0.3	106149
-0.3 to -0.2	121799
-0.2 to -0.1	5303
-0.1 to 0.0	2749
0.0 to 0.1	24630
0.1 to 0.2	7744
0.2 to 0.3	2452
0.3 to 0.4	2464
0.4 to 0.5	10303
0.5 to 0.6	12340
0.6 to 0.7	15420
0.7 to 0.8	16822
0.8 to 0.9	22215
0.9 to 1.0	138263

For the first graph, a high similarity threshold (cosine similarity > 0.9) was used. The statistics for the distances between city pairs in this graph are as follows:

- *Mean distance*: 7,942.658 km
- *Median distance*: 7,741.326 km
- *Standard deviation*: 5,129.801 km
- *Minimum distance*: 1.932 km
- *Maximum distance*: 19,975.287 km

The shortest distance is between Jerusalem, Israel (latitude 31.7784, longitude 35.2066), and Al Quds, West Bank (latitude 31.7764, longitude 35.2269), which are geographically very close, underscoring their proximity with nearly identical latitude and longitude coordinates. The longest distance is between Quito, Ecuador (latitude -0.2150, longitude -78.5001), and Pekanbaru, Indonesia (latitude 0.5650, longitude 101.4250), which are on opposite sides of the globe, as reflected by their significantly contrasting geographical coordinates.

For the second graph, which was defined by a cosine similarity threshold ranging from -0.4 to -0.2, a moderate level of climatic similarity was observed among city pairs. The key statistics for this graph are as follows:

- *Mean distance*: 8,648.245 km
- *Median distance*: 8,409.507 km
- *Standard deviation*: 4,221.592 km
- *Minimum distance*: 115.137 km
- *Maximum distance*: 19,963.729 km

In this graph, the shortest distance is between Kabul, Afghanistan (latitude 34.5167, longitude 69.1833), and Jalalabad, Afghanistan (latitude 34.4415, longitude 70.4361). The longest distance is between Mendoza, Argentina (latitude -32.8833, longitude -68.8166), and Shiyan, China (latitude 32.5700, longitude 110.7800).

Table 4. Cities in the Third Connected Component (7 Nodes)

City	Country	Latitude	Longitude	Label
Jinhua	China	29.12	119.65	0
Quetta	Pakistan	30.22	67.03	0
Xiantao	China	30.37	113.44	0
Jingling	China	30.65	113.16	0
Xiaoxita	China	30.70	111.28	1
Kandahar	Afghanistan	31.61	65.69	1
Saidu Sharif	Pakistan	34.75	72.35	1

Table 5. Cities in the Fourth Connected Component (5 Nodes)

City	Country	Latitude	Longitude	Label
Agadir	Morocco	30.44	-9.62	0
Alexandria	Egypt	31.20	29.95	1
Port Said	Egypt	31.26	32.29	1
Benghazi	Libya	32.17	20.07	1
Tijuana	Mexico	32.50	-117.08	1

For the third graph, we applied a very high similarity threshold (cosine similarity > 0.99), resulting in connected components of sizes [514, 468, 7, 5]. The largest connected component, with 514 nodes, predominantly consists of cities with stable climates (475 nodes labeled as stable) and a smaller portion with unstable climates (39 nodes labeled as unstable). The second-largest component, containing 468 nodes, primarily consists of cities with unstable climates (451 nodes labeled as unstable) and a few with stable climates (17 nodes labeled as stable). These findings suggest that cities within the same climate category (stable or unstable) exhibit higher similarity, forming larger connected components, while similarities across different climate categories are less pronounced.

The smaller connected components represent city graphs located at the transition zone between stable and unstable climates. Tables 4 and 5 provide details of these cities, highlighting their transitional nature. Table 4 lists the cities in the third component, consisting of 7 nodes, while Table 5 lists the cities in the fourth component, comprising 5 nodes. These cities demonstrate the variability and complexity of climatic relationships, showing a mix of stable and unstable

conditions. This highlights the nuanced and intricate climatic patterns that exist at the boundaries between different climate categories.

5 Conclusion

In this study, we demonstrated the effectiveness of using pre-final vectors from GNN Graph Classification models to enhance climate data analysis. By embedding entire small graphs within these models, we introduced a novel approach to exploring complex climatic patterns and assessing city-level temperature stability. Our methodology enabled tasks such as nearest neighbor identification, clustering, and meta-graph construction, providing deeper insights into climate dynamics. The high classification accuracy achieved in our experiments demonstrates the capability of GNNs to handle intricate graph structures, showcasing their potential in distinguishing between different data patterns, such as stable and unstable climate profiles.

Additionally, the integration of linear algebra techniques with GNN Graph Classification models proved invaluable for managing and interpreting diverse datasets, underscoring the importance of advanced mathematical frameworks in deep learning and AI. This combination allows for a more nuanced understanding of complex data structures, enhancing the ability to extract meaningful patterns and relationships from vast datasets.

Looking ahead, the integration of GNN Graph Classification models with pre-final vector analysis opens up promising avenues for further research. Refining graph labeling methods by incorporating additional environmental factors, such as atmospheric conditions, hydrology, and oceanography, could provide a more comprehensive understanding of environmental dynamics. Beyond environmental science, pre-final vectors have potential applications in other domains, including finance, neuroscience, social network analysis, and natural language processing. By continuing to explore these applications, we can leverage GNNs to uncover new insights and enhance our understanding of complex data environments across various fields.

References

1. Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press, (2016).
2. Sutskever, I., Martens, J., Dahl, G., & Hinton, G. On the importance of initialization and momentum in deep learning. *Proceedings of the 30th International Conference on Machine Learning, 28*, 1139-1147, (2013).
3. something2vec: Overview of various *2vec models, Available at: https://gist.github.com/nzw0301/333afc00bd508501268fa7bf40cafe4e, Last accessed: 2024.
4. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I.: Attention Is All You Need. (2017).
5. Deep Graph Library (DGL): Link Prediction using Graph Neural Networks. Available online: https://www.dgl.ai (2018).

6. Romanova, A. Rewiring Knowledge Graphs by Graph Neural Network Link Predictions. In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence.*(2023).
7. Romanova, A.: Uncovering Hidden Connections: Granular Relationship Analysis in Knowledge Graphs. In: Proceedings of International Conference on Information Technology and Applications. (2024).
8. Romanova, A. GNN Graph Classification Method to Discover Climate Change Patterns. In: Artificial Neural Networks and Machine Learning – ICANN 2023. Lecture Notes in Computer Science, vol 14257. Springer, Cham. https://doi.org/10.1007/978-3-031-44216-2-32 (2023)
9. kaggle.com, "Temperature History of 1000 cities 1980 to 2020," (2020).
10. Romanova, A. Enhancing Time Series Analysis with GNN Graph Classification Models. In Complex Networks & Their Applications XII. DOI: 10.1007/978-3-031-53468-3-3, (2024).
11. Romanova, A. Enhancing NLP through GNN-Driven Knowledge Graph Rewiring and Document Classification. In Proceedings of the 2024 35th Conference of Open Innovations Association (FRUCT). DOI: 10.23919/FRUCT61870.2024.10516410, (2024).
12. A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, (2012).
13. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, (2015).
14. N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor, "Industry-scale Knowledge Graphs: Lessons and Challenges," *acmqueue*, (2019).
15. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv*, doi=10.48550/arXiv.2104.13478. (2021).
16. Adamczyk, J.: Application of Graph Neural Networks and graph descriptors for graph classification. (2022).
17. He, H., Queen, O., Koker, T., Cuevas, C., Tsiligkaridis, T., Zitnik, M.: Domain Adaptation for Time Series Under Feature and Label Shifts. (2023).
18. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for Pre-training Graph Neural Networks. (2020).
19. Liu, X., Chen, J., Wen, Q.: A Survey on Graph Classification and Link Prediction based on GNN. arXiv:2307.00865 [cs.LG] (2023).
20. Xu, N., Kosma, C., Vazirgiannis, M.: TimeGNN: Temporal Dynamic Graph Learning for Time Series Forecasting. arXiv:2307.14680 [cs.LG] (2023).
21. Ardabili, S., Mosavi, A., Dehghani, M., Varkonyi-Koczy, A.: Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets. (2019).
22. Liu, Y., Racah, E., Correa, J., Khosrowshahi, A., Lavers, D., Kunkel, K., Wehner, M., Collins, W.: Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets. (2016).
23. Liu, Y., Racah, E.: Deep Learning and Machine Learning in Hydrological Processes, Climate Change and Earth. (2019).
24. PyG, "Pytorch Geometric Library: Graph Classification with Graph Neural Networks," (2023).
25. sparklingdataocean.com. "Vectors in Graph Neural Networks: Detailed Analysis and Coding". [Online]. Available: http://sparklingdataocean.com/2024/07/04/vectorsGNN/. (2024)