sciendo

CAIM
ISSN 2038-0909

Commun. Appl. Ind. Math. 15 (2), 2024, 91–101 DOI: 10.2478/caim-2024-0015

Research Article

Learning stability on graphs

Antonioreneé Barletta[1*], Salvatore Cuomo[2*], Gianluca Milano[1 *]

[1]Fabritech Division, Spici srl, Naples, Italy

[2]Department of Mathematics and application "Renato Caccioppoli", University of Naples "Federico II", Naples, Italy

[*]Email address for correspondence: barletta@spici.eu

Communicated by Clemente Cesarano

Received on 07 10, 2020. Accepted on 09 30, 2024.

Abstract

In artificial intelligence applications, the model training phase is critical and computationally demanding. In the graph neural networks (GNNs) research field, it is interesting to investigate how varying the graph topological and spectral structure impacts the learning process and overall GNN performance. In this work, we aim to theoretically investigate how the topology and the spectrum of a graph changes when nodes and edges are added or removed. Numerical results highlight stability issues in the learning process on graphs. In this work, we aim to theoretically investigate how the topology and the spectrum of a graph changes when nodes and edges are added or removed. We propose the topological relevance function as a novel method to quantify the stability of graph-based neural networks when graph structures are perturbed. We also explore the relationship between this topological relevance function, Graph Edit Distance, and spectral similarity. Numerical results highlight stability issues in the learning process on graphs.

Keywords: `Scientific Machine Learning, Graphs, Stability, Learning Process.`

AMS subject classification: `05C85, 65Z05, 68R10, 68W01.`

1. Introduction

Many real-world applications are based on machine and deep learning tools. As the deployment of these models becomes more pervasive, the importance of robustness in Artificial Intelligence (AI) systems has emerged as a central area of focus. Robustness refers to the ability of a machine learning model to maintain its performance and functionality under a variety of conditions, including the presence of noise, and variations in the input data. Despite the advances in model accuracy, many state-of-the-art AI models remain vulnerable to adversarial perturbations and other forms of data manipulation, for example, in the cybersecurity scenario [1]. A crucial aspect of the effective use of these methodologies is the learning process, which is heavily dependent on the model structure [2], and also by the quantity and quality of the data used during the training phase [3]. Generally, the robustness of AI models is measured in terms of convergence and stability of the learning process. Convergence refers to the process during training whereby the model's parameters stabilize, and the loss function reaches a minimum or a sufficiently low value. This indicates that the model has effectively learned the underlying patterns in the training data, optimizing its performance according to the chosen loss function. Stability refers to the ability of a trained model to continue to deliver valid results even when the input data is perturbed. This ensures that the model maintains high performance and accuracy despite variations or noise in the data. In recent years, there has been an ever-increasing interest in AI models based on graphs. These structures are capable of representing and describing many scenarios, such as social networks, traffic management and forecasting, molecule depiction and interaction, and cybersecurity, just to mention a few of them. Ensuring the robustness and stability of these models is crucial, especially in dynamic environments where graphs are frequently modified. In this paper, we focus on quantifying models stability on graphs using topological information and spectral properties. Moreover, we present some initial results about the stability of the learning process on graph-based AI models.

The paper is structured as follows; in section 2 we provide a mathematical introduction to graphs, in section 3 a formal methodology to manipulate a graph is presented, in terms of both edge and node modification. Subsequently, in section 4 we start by introducing the graph edit distance, a classical parameter used in literature for comparing two graphs, highlighting its main advantage and criticality. Later, we present two new similarity indicators, the spectral distance which works on the spectral domain, and the topological relevance function defined in terms of topological quantities. We then present in section 5 some numerical results on the use of both the topological relevance function and the spectral distance. Lastly, in section 6 we outline some conclusions.

2. Mathematical Background

We start with some useful definitions

Definition 2.1. We define a graph as an ordered pair $G = (V, E)$ where V is the set containing the vertices (also known as nodes), and E is the set containing the edges (also known as links). To avoid ambiguity we will refer to such an object as an *undirected graph*

To every graph, G corresponds a matrix called *Adjacency matrix* defined as

$$A_{ij} \coloneqq \begin{cases} w_{ij} & \text{if there is an edge between vertex } v_i \text{ and vertex } v_j, \\ 0 & \text{otherwise,} \end{cases}$$

where $w_{ij} \in \mathbf{R}^+$ is the weight of the edge connecting the vertices v_i and v_j. For an unweighted graph w_{ij} is always 1.

Another important element in the description of a graph G is the *degree matrix*

$$D_{ij} \coloneqq \begin{cases} \sum_k w_{ik} & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

Thus, each diagonal element D_{ii} of the degree matrix D represents the sum of the weights of all edges incident to the vertex v_i, and all off-diagonal elements are zero. Clearly, for unweighted graphs, the element D_{ii} is the total number of edges that start from vertex v_i.

Definition 2.2. The *Laplacian matrix* L of the graph G is defined in terms of the degree matrix D and the adjacency matrix A. The Laplacian matrix is given by:

$$L = D - A,$$

the entries of the Laplacian matrix are thus defined as follows:

$$L_{ij} = \begin{cases} \sum_k w_{ik} & \text{if } i = j, \\ -w_{ij} & \text{if there is an edge between vertex } v_i \text{ and vertex } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

Often in the context of spectral graph theory, machine learning, and data analysis instead of the standard graph Laplacian matrix, it is preferred to use its *normalized* version

$$\mathcal{L} \coloneqq D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}.$$

Normalization ensures that the graph Laplacian accounts for the degree of each vertex, making the Laplacian scale invariant. This is particularly useful when dealing with graphs where the vertex degrees vary significantly. The normalized Laplacian helps to balance the influence of nodes with different degrees.

We recall that the spectrum of a matrix is the ordered set of its eigenvalues. We denote by λ_k the k^{th} eigenvalue of the normalized Laplacian matrix $\mathcal{L}$, for which we choose an ascending order

$$\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_n. \tag{1}$$

The following properties related to the spectrum of $\mathcal{L}$ are true

- the eigenvalues of $\mathcal{L}$ lie in the range $[0, 2]$ [4],
- the smallest eigenvalue λ_0 is 0, and its multiplicity corresponds to the number of connected components in the graph [4],
- for connected graph, λ_1, i.e., the smallest non-zero eigenvalue, represents an important parameter and can provide information about the connectivity and expansion properties of the graph. [5,6]

A graph is completely described by three aspects: its *topological structure* retaining information about the number of nodes and edges; *the Laplace matrix spectrum*, which contains information on the weight of the edges; and lastly, the node weight for which it does not exist a general structure.

3. How to manipulate a graph

Inspired by the work of Pineau [7], we present a formal method to modify a graph. The operation of adding or removing edges, as well as modifying edge weights, can be performed straightforwardly by defining an $n \times n$ symmetric matrix P, where n is the number of nodes of the graph. P has to satisfy two properties

i) $P_{ii} = 0 \quad \forall i = 1, \cdots, n$,
ii) $P_{ij} \in [-w_{ij}, +\infty)$ such that $A_{ij} + P_{ij} = \widetilde{w}_{ij} \in [0, +\infty)$,

where $\widetilde{w}_{ij}$ is the new weight of the edge connecting the vertices v_i and v_j. By adding this matrix to the adjacency matrix A of the graph G

$$\widetilde{A} = A + P,$$

we can add or remove edges and also change the weights of existing ones, where $\widetilde{A}$ is the adjacency matrix of the new graph $\widetilde{G}$. The new degree matrix $\widetilde{D}$ can be obtained trough the matrix P as:

$$\widetilde{D}_{ij} = \begin{cases} D_{ij} + \sum_k P_{ik} & \text{if } i \neq j, \\ 0 & \text{if } i = j. \end{cases} \tag{2}$$

As stated in [7], matrices of the form $P = -A + \pi A \pi$, where π is a permutation matrix, are not interesting in terms of edge modification as they reduce to a simple index permutation.

Adding isolated nodes to a graph can be performed by extending the adjacency matrix according to the following scheme

$$\overline{A} = \begin{bmatrix} A & O_{n\times m} \\ O_{m\times n} & O_{m\times m} \end{bmatrix},$$

here A is the adjacency matrix of the original graph G, $O_{n\times m}$, $O_{m\times n}$, $O_{m\times m}$ are null matrices, and $\overline{A}$ is the adjacency matrix of the extended graph $\overline{G}$. Consequently, the degree matrix will be extended as

$$\overline{D} = \begin{bmatrix} D & O_{n\times m} \\ O_{m\times n} & O_{m\times m} \end{bmatrix}.$$

Remark 3.1. From a spectral point of view, adding m isolated nodes to a graph only increases the algebraic multiplicity of the null eigenvalue λ_0 by m, without any modification to the non-null eigenvalues.

Proof. The normalized Laplacian of the extended graph is $\overline{\mathcal{L}} = \overline{D}^{-1/2} \overline{L}\, \overline{D}^{-1/2}$, where $\overline{L} = \overline{D} - \overline{A}$. We point out that $\overline{D}^{-1/2}$ is the square root of the pseudo-inverse of $\overline{D}$. By explicitly writing the normalized

Laplacian

$$\overline{\mathcal{L}} = \overline{D}^{-1/2}\overline{D}\,\overline{D}^{-1/2} - \overline{D}^{-1/2}\overline{A}\,\overline{D}^{-1/2} = \bar{I} - \overline{D}^{-1/2}\overline{A}\,\overline{D}^{-1/2} \tag{3}$$

where:

$$\bar{I} \coloneqq \overline{D}^{-1/2}\overline{D}\,\overline{D}^{-1/2} = \begin{bmatrix} I & O_{n\times m} \\ O_{m\times n} & O_{m\times m} \end{bmatrix}, \tag{4}$$

and I_n is the identity matrix of size n. We can therefore write

$$\det(\overline{\mathcal{L}} - \lambda I_{n+m}) = 0 \quad \Longrightarrow \quad \det\begin{pmatrix} \mathcal{L} - I_n\lambda & O_{n\times m} \\ O_{m\times n} & \lambda I_m \end{pmatrix} = 0, \tag{5}$$

that correspond to

$$\lambda^m \cdot \det(\mathcal{L} - I_n\lambda) = 0. \tag{6}$$

□

For the removal of nodes, we start by observing that node index permutations do not change the graph structure, i.e., the graph topology is invariant under node index permutations. Therefore, we can always find a permutation π to apply to the adjacency matrix A, such that the nodes to remove are represented by the last m rows and m columns of A. Thus, the adjacency matrix of the reduced graph can be written as

$$\underline{A} = \begin{bmatrix} A_{[n-m,n-m]} & O_{(n-m)\times m} \\ O_{m\times(n-m)} & O_{m\times m} \end{bmatrix},$$

where $A_{[n-m,n-m]}$ is a sub-matrix consisting of the first $n-m$ rows and $n-m$ columns of A. The corresponding degree matrix is

$$\underline{D} = \begin{bmatrix} D_{[n-m,n-m]} & O_{(n-m)\times m} \\ O_{m\times(n-m)} & O_{m\times m} \end{bmatrix},$$

here $D_{[n-m,n-m]}$ is the corresponding sub-matrix of D.

For the last point, concerning the modification of node weights, to the best of our knowledge, there is no general methodology to vary the weights of the nodes except for a direct node-by-node modification.

Numerical example of graph modification

Let us consider an undirected weighted graph G (See Fig. 1), with the following adjacency matrix A:

$$A = \begin{pmatrix} 0 & 1.5 & 2.1 & 0 \\ 1.5 & 0 & 0.8 & 1.35 \\ 2.1 & 0.8 & 0 & 0 \\ 0 & 1.35 & 0 & 0 \end{pmatrix}$$

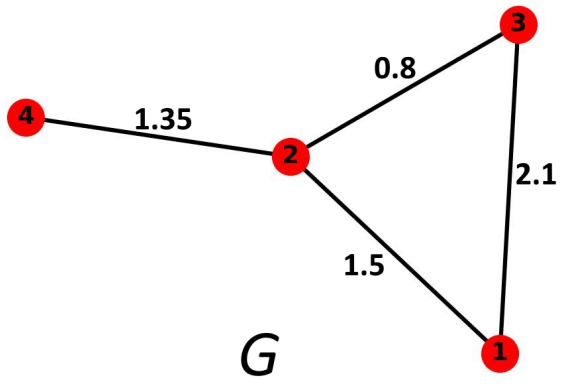

Figure 1. Image depicting a weighted undirected graph. The numbers on the edges represent their corresponding weights.

The addition of an isolated node, in the formalism of this work, can be achieved by extending the adjacency matrix, such that

$$\overline{A} = \begin{bmatrix} A & O_{3\times 1} \\ O_{1\times 3} & O_{1\times 1} \end{bmatrix} = \begin{pmatrix} 0 & 1.5 & 2.1 & 0 & 0 \\ 1.5 & 0 & 0.8 & 1.35 & 0 \\ 2.1 & 0.8 & 0 & 0 & 0 \\ 0 & 1.35 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In order to add, remove or modify existing edges, we define a matrix P

$$P = \begin{pmatrix} 0 & -1.5 & -0.45 & 0 & 0 \\ -1.5 & 0 & 0 & 0.25 & 0 \\ -0.45 & 0 & 0 & 0 & 1 \\ 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

the new adjacency matrix is

$$\widetilde{A} = \overline{A} + P = \begin{pmatrix} 0 & 0 & 1.65 & 0 & 0 \\ 0 & 0 & 0.8 & 1.6 & 0 \\ 1.65 & 0.8 & 0 & 0 & 1 \\ 0 & 1.6 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

A graphical representation of all the previous operations is given by Figure 2.

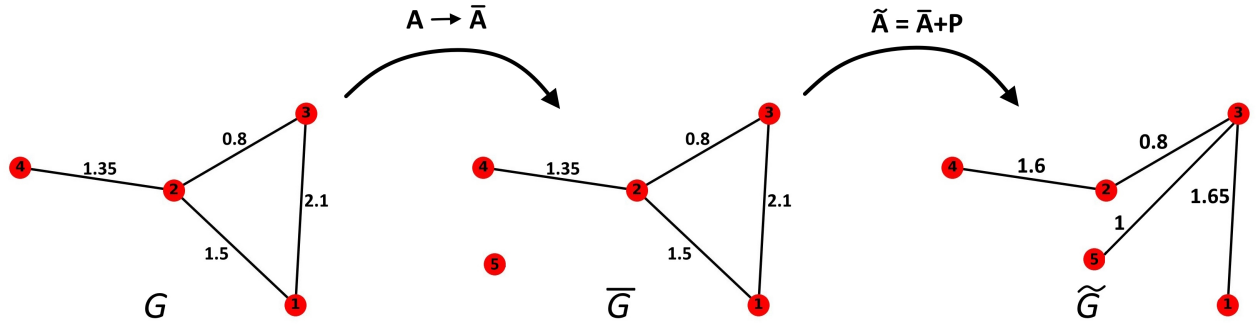

Figure 2. Pictured representation of graph manipulations.

4. Similarities of graphs

One of the main problems in Graph Machine Learning is that when a model is trained on a specific graph, and this graph is later modified, the model needs to be retrained, resulting in both a cost in terms of time and energy. A preliminary step in studying the learning process is to identify a measure of closeness between the original graph G_1 and the modified graph G_2.

In scientific literature, many possible ways to compare two graphs are described. A very important one is *graph edit distance* (GED), defined as the minimum cost of transforming one graph into another through a sequence of edit operations, which include insertion, substitution, and removal of nodes and edges. Each of these edit operations has an assigned cost, typically determined based on the specific application, and can be tailored to reflect the significance of different types of edits [8]. Mathematically the GED between a graph G_1 and G_2 can be expressed as

$$\mathrm{GED}(G_1, G_2) = \min_{(e_1,\cdots,e_k)\in\mathcal{T}} \sum_{i=1}^{k} c(e_i), \tag{7}$$

where $\mathcal{T}$ is the set of all possible edit paths, i.e., sequences of edit operations that transform G_1 in G_2, e_k are the single edit operation, and $c(e_i)$ is the cost associated with the edit operation e_i.

GED can be customized for different applications by defining various cost functions for edit operations, making it theoretically versatile for many types of graph data. However, the value of GED is highly dependent on the choice of these cost functions; poorly chosen cost functions can result in misleading similarity measures. Furthermore, calculating the exact GED between two graphs is an NP-hard problem. As a result, several efficient approximation algorithms have been developed to tackle this challenge. Most of these approximation algorithms have a cubic computational time complexity, as stated in [9].

Another similarity indicator between the two graphs is related to their normalized Laplacian spectrum. The closeness of the spectra can be evaluated by comparing the eigenvalues of the Laplacian matrices of the two graphs. If the graphs are of different order to preserve matrix dimensionality, we may need to add isolated nodes to the smaller graph. As already stated, adding isolated nodes does not perturb the non-null spectrum, as it will only increase the algebraic multiplicity of the eigenvalue $\lambda_0 = 0$. We will define the *spectral similarity* parameter as the ℓ^1 norm of the difference between the spectra, namely:

$$S(G_1, G_2) \coloneqq \sum_{i=1}^{n} \left| \lambda_i^{G_1} - \lambda_i^{G_2} \right|. \tag{8}$$

One key advantage of spectral similarity is that the spectrum naturally encodes information about the edges of a graph and their respective weights. Furthermore, this parameter is less misleading compared to GED, as it does not depend on the choice of a cost function. However, two graphs with identical spectra are not necessarily isomorphic, meaning they can have different structures [10]. This limitation highlights potential ambiguities in spectral graph theory, as different graph topologies can produce the same spectral properties, complicating applications that rely solely on eigenvalue comparisons.

In this paper, we introduce a new topological feature for graphs called "Topological Relevance Function" (TRF), defined as

$$\begin{aligned} \mathrm{TRF}(G) \coloneqq 1 + \mu_G \Bigg[& w_A \left(\frac{n - A}{A} \right) + w_{N_c} \left(\frac{n - 1 - N_c}{N_c} \right) \\ & + w_{E_c} \left(\frac{n - 1 - E_c}{E_c} \right) + w_E \left(\frac{1 - E}{E} \right) + w_L \left(L - 1 \right) \Bigg]. \end{aligned} \tag{9}$$

Here, μ_G is a multiplicative factor that serves to penalize or reward the global structure of the TRF defined as

$$\mu_G = w_\mu \cdot \frac{\mathrm{Card}(E)}{n},$$

where $\mathrm{Card}(E) = m$ is the number of edges and w_μ a weight to further penalize that ratio. The other quantities are:

- n the *number of nodes* of the graph,
- A the *algebraic connectivity*,
- N_c the *node connectivity*,
- E_c the *edge connectivity*,
- E the *global efficiency* of the graph,
- L the *average path length*,

while w_A, w_{N_c}, w_{E_c}, w_E, and w_L are positive real numbers and serve as weights of the TRF.

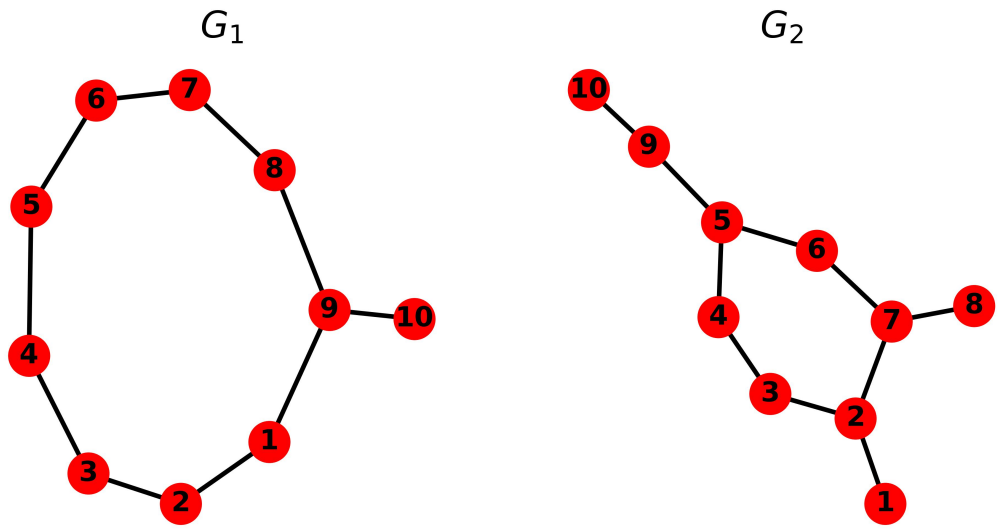

Figure 3. Two example of undirected, unweighted graphs with ten edges and ten nodes. Note that the two graphs have a different topological structure.

In table (Tab. 4) we evaluated the TRF of G_1 and G_2, and also evaluated the TRF while setting w_A to zero

Quantity	G_1	G_2	Notes
TRF	47.3218	55.4709	lower is better
TRF with $w_A = 0$	19.6609	19.5766	lower is better
A	0.3489	0.2710	higher is better
N_c	1	1	higher is better
E_c	1	1	higher is better
E	0.4959	0.5059	higher is better
L	2.644	2.6	lower is better

The TRF presented in Eq. (9), for trivial graphs, i.e. graphs that contains a single node, cannot be used as a topological indicator. Furthermore, it is well-defined only for graphs with a single connected component. However, in many real-world scenarios, graph datasets consist of multiple connected components and to address this issue, we define a generalized version of the TRF as:

$$\mathrm{TRF}_{\mathrm{Gen}}(G) := \mu_G \cdot \alpha_G \cdot \sum_{i=1}^{k_1} \mathrm{TRF}(G_i) + k_2 \cdot w_{\mathrm{iso}}, \tag{10}$$

where, μ_G and α_G are two penalizing factors given by

$$\alpha_G := 1 + w_{cc} \cdot \frac{k-1}{n}, \tag{11}$$

$$\mu_G := \begin{cases} 1 & \text{if } k = 1, \\ w_\mu \cdot \dfrac{m}{n} & \text{otherwise.} \end{cases} \tag{12}$$

The quantities in Eqs. (10), (11), (12) are

- k the total number of connected components,
- k_1 the number of non-trivial connected components,
- $k_2 = k - k_1$ the number of isolated nodes,
- w_{iso} a weight to assign to isolated nodes,
- w_{cc} a weight to penalize the presence of connected components,
- G_i is the i-th non trivial connected subgraph of G.

From now on we will refer to the generalized TRF also as TRF or GTRF. It is important to note that, with this definition, if G has a single connected component (excluding the trivial graph consisting of a single node), we retrieve Eq. (9). Moreover, unlike in the case of connected graphs, isolated nodes are topologically significant when the graph has multiple connected components.

This TRF or its generalized version can be used to compare the topological structure of graphs. Close TRF values are synonymous of topological similarity, conversely the higher the "distance", the higher is the difference between the graphs structure. In this section, we presented three approaches of how to measure graph similarity: GED, TRF, and GTRF. Each of these indicators served different purposes and presented unique advantages and criticalities, depending on the type of graph and the computational resources available. Table 4 summarizes the key aspects of the aforementioned indicators.

Metric	Focus	Strengths	Weaknesses	Best Used For
GED	Edit distance between graphs	Direct, intuitive measure of graph modifications	NP-hard, sensitive to cost functions	Small graphs, pattern recognition, applications needing detailed graph transformations
TRF	Global topological structure	Captures overall graph properties, computationally efficient	Not suitable for disconnected graphs, lacks fine-grained edit details	Network analysis, biological systems, where topology matters
GTRF	Global topological structure with multiple components	Handles disconnected graphs, more generalized than TRF	Still lacks fine-grained details, depends on penalizing factors	Real-world dynamic graphs with isolated nodes or multiple components

5. Numerical Results

We start by comparing the topological and spectral structure of a graph G_1 whose corresponding topological relevance function value is $TRF(G_1) = 54.3537$. By removing the edge connecting node 1 and node 9 a new graph, as depicted in Fig. 4.

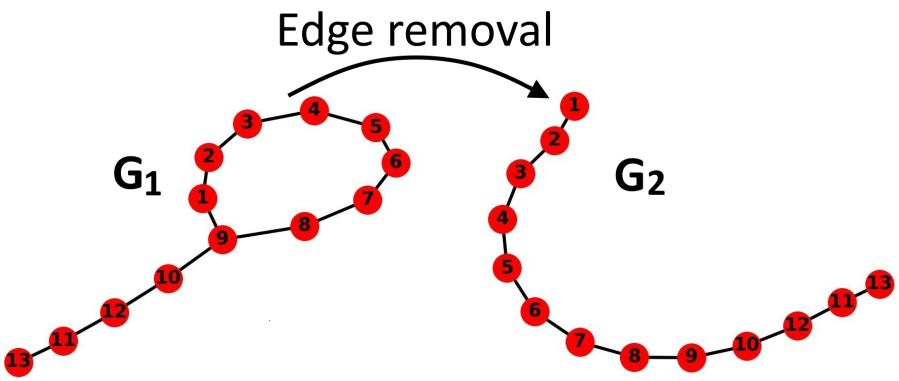

Figure 4. The effect of edge removal on graph G_1. The resulting graph G_2 becomes a path graph, which has a worst overall connectivity structure compared to G_1.

By evaluating the TRF of the perturbed graph G_2, where $TRF(G_2) = 79.5670$, we can observe that its value has increased, thus resulting in an overall worst topological structure in terms of connectivity. We also computed the spectral distance $S(G_1, G_2) = 0.6879$ between G_1 and G_2. The graph G_1 was also modified by removing the node 9, resulting in a complete change in terms of topological structure as shown in Fig. 5. This topological modification is well captured by the TRF value $TRF(G_3) = 122.4784$ of G_3, and the spectral distance between G_1 and G_3 is $S(G_1, G_3) = 1.3317$. We notice that also from the spectral perspective, graph G_3 has a greater distance compared to graph G_2.

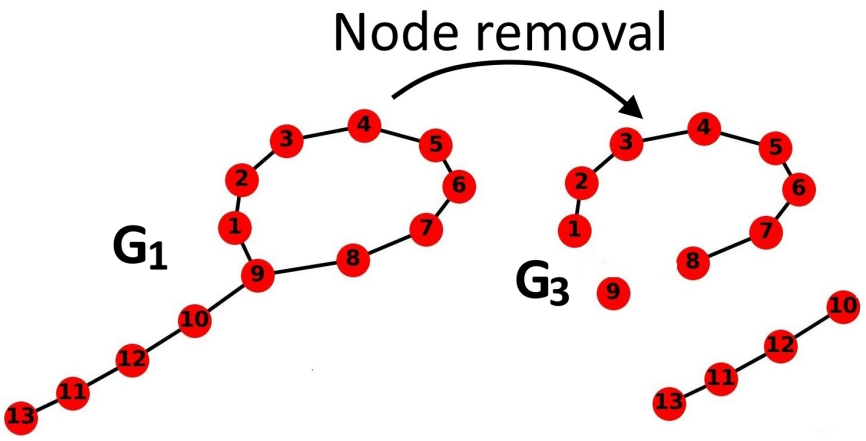

Figure 5. The effect of node removal on graph G_1. The new graph G_3 consists of two path graphs and an isolated node. We highlight that in our framework, removing a node corresponds to eliminating every edge connected to the node.

To study the stability of the learning process on graphs, we tackle the classification problem of the Protein dataset [11]. This dataset is a collection of 1113 graphs representing proteins, where the nodes correspond to amino acids and two nodes are connected by an edge if they are less than 6 Angstroms apart. Each graph has a label (0 or 1) if it is an enzyme or not. To classify these proteins we used a graph convolutional neural network (GCN) with this structure:

```
   GCNN(
(conv1): GCNConv(3, 12)
(conv2): GCNConv(12, 24)
(conv3): GCNConv(24, 48)
(conv4): GCNConv(48, 24)
(conv5): GCNConv(24, 6)
(fc): Linear(in_features=6, out_features=2, bias=True)
).
```

The dataset was then split with an 80-10-10 ratio for the train, validation, and test set, respectively. We trained the model using the train set and evaluated its performance on the test set. The test dataset was slightly perturbed by adding or removing an edge from each of the proteins. The network outputs are outlined in Table 3.

	TS GTRF= 28039				**TS GTRF= 27649**			
	Prec.	**Recall**	**F1-Score**	**Supp.**	**Prec.**	**Recall**	**F1-Score**	**Supp.**
0	0.73	0.77	0.75	70	0.75	0.74	0.75	70
1	0.58	0.52	0.55	42	0.58	0.60	0.59	42
W. avg.	0.67	0.68	0.68	112	0.69	0.69	0.69	112

Note that the corresponding metrics for the test set and its perturbed version are similar; this is indicated by the fact that the GTRF values are close. Table 4 represents the confusion matrix of the model.

	Test Set		**Perturbed Test Set**	
	Pred 0	**Pred 1**	**Pred 0**	**Pred 1**
Actual 0	54	16	52	18
Actual 1	20	22	17	25

We selected two specific proteins and their corresponding perturbed version, to show the stability properties of the learning process.

	Unstable	**Stable**
TRF	16.075	75.686
Perturbed TRF	12.056	69.665
% TRF difference	24.995%	7.954%
Spectral Distance	1.438	1.082
y true	1	0
y pred.	1	0
y pred. perturbed	0	0

6. Conclusions

Despite the staticity of mathematical graphs, in many real-world applications, graphs are dynamic entities where edges and nodes are added or removed, and weights are modified. AI models require an expensive training phase, in terms of both computational resources and time. Furthermore, the quality of the learning process is heavily dependent on the training phase. To use correctly pre-trained models on modified graphs, it is necessary to have stability indicators for the learning process. In this paper, we suggest searching those indicators both in the spectral domain and in the topological structure of a graph. Initial numerical investigations indicate that the two similarity parameters can be used to evaluate the stability of the learning process. Our future research aim is to find a mathematical correlation between graph similarities and the predicted results of a GNN, or more general AI model based on graphs.

Acknowledgments

A. Barletta and G. Milano have been partially supported by the Italian project CTE - "Casa delle Tecnologie Emergenti di Napoli" (CUP: B67F23000000008) funded by ministero delle imprese e del made in Italy (MIMIT) and project "FANTAS-IA" funded by the "Casa delle Tecnologie Emergenti di Matera" (CUP: I14E20000020001) under the CALL4future experimentation project.

S. Cuomo also acknowledges GNCS-INdAM and the UMI-TAA, UMI-AI research groups. S. Cuomo has been partially supported by the Italian PRIN project "Numerical Optimization with Adap-

tive accuracy and applications to machine learning" (CUP: E53D23007690006), Progetti di Ricerca di Interesse Nazionale 2022 and PNRR Centro Nazionale HPC, Big Data e Quantum Computing, (CN_00000013)(CUP: E63C22000980007), under the NRRP MUR program funded by the NextGenerationEU.

References

1. L. Sun, M. Tan, and Z. Zhou, A survey of practical adversarial example attacks, *Cybersecurity*, vol. 1, 12 2018.
2. L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-dujaili, Y. Duan, O. Al-Shamma, J. I. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, Review of deep learning: concepts, cnn architectures, challenges, applications, future directions, *Journal of Big Data*, vol. 8, 2021.
3. L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, The effects of data quality on machine learning performance, 2022.
4. F. Chung, *Spectral Graph Theory*. No. No. 92 in CBMS Regional Conference Series, Conference Board of the Mathematical Sciences, 1996.
5. M. Fiedler, Algebraic connectivity of graphs, *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.
6. M. Fiedler, Laplacian of graphs and algebraic connectivity, *Banach Center Publications*, vol. 25, no. 1, pp. 57–70, 1989.
7. E. Pineau, Using laplacian spectrum as graph feature representation, 2019.
8. A. Sanfeliu and K.-S. Fu, A distance measure between attributed relational graphs for pattern recognition, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, pp. 353–362, 5 1983.
9. Z. Zeng, A. Tung, J. Wang, J. Feng, and L. Zhou, Comparing stars: On approximating graph edit distance., *PVLDB*, vol. 2, pp. 25–36, 01 2009.
10. S. Butler and J. Grout, A construction of cospectral graphs for the normalized laplacian, 2012.
11. K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, Protein function prediction via graph kernels, *Bioinformatics*, vol. 21, pp. i47–i56, 06 2005.