MARA: a deep learning based framework for multilayer graph simplification

Cheick Tidiane Ba[a,e,*], Roberto Interdonato[b,d], Dino Ienco[c,d], Sabrina Gaito[a]

[a]*Department of Computer Science, University of Milan, Milan, Italy*
[b]*CIRAD, UMR TETIS, Montpellier, France*
[c]*INRAE, UMR TETIS, Univ. Montpellier, Montpellier, France*
[d]*INRIA, Montpellier, France*
[e]*Queen Mary University of London, London, United Kingdom*

Abstract

In many scientific fields, complex systems are characterized by a multitude of heterogeneous interactions/relationships that are challenging to model. Multilayer graphs constitute valuable tools that can represent such complex systems, thus making possible their analysis for downstream decision-making processes. Nevertheless , modeling such complex information still remains challenging in real-world scenarios . On the one hand, holistically including all relationships may lead to noisy or computationally intensive graphs. On the other hand, limiting the amount of information to model through the selection of a portion of the available relationships can introduce boundary specification biases. However, the current research studies are demonstrating that it is more beneficial to retain as much information as possible and at a later stage perform graph simplification i.e., removing uninformative or redundant parts of the graph to facilitate the final analysis. While simplification strategies, based on deep learning methods, have been already extensively explored in the context of single-layer graphs, only a limited amount of efforts have been devoted to simplification strategies for multilayer graphs. In this work, we propose the MultilAyer gRaph simplificAtion (MARA) framework, a GNN-based approach designed to simplify multilayer graphs based on the downstream task. MARA generates node embeddings for a specific task by training jointly two main components: i) an edge simplification module and ii) a (multilayer) graph neural network. We tested MARA on different real-world multilayer graphs for node classification tasks. Experimental results show the effectiveness of the proposed approach: MARA reduces the dimension of the input graph while keeping and even improving the performance of node classification tasks in different domains and across graphs characterized by different structures. Moreover, deep learning-based simplification allows MARA to preserve and enhance important graph properties for the downstream task. To our knowledge, MARA represents the first simplification framework especially tailored for multilayer graphs analysis .

Keywords: graph neural network, graph simplification, multilayer graph

1. Introduction

The graph analysis and mining research field has raised in popularity in the last two decades, thanks to the ability of graphs to model a wide range of real-life phenomena from physical [1] to biological [2] and social systems [3], from scientific [4] to financial data [5, 6], transportation routes [7], and many others [8]. In this regard, the multilayer graph model [9] is widely used as a powerful tool to represent the organization and relationships of complex systems covering many different domains. Graphs serve as models for the relationships among interconnected entities, usually depicted as nodes (or vertices) linked by edges (or links) symbolizing interactions or dependencies. Multilayer graphs extend the graph model, allowing the definition of layers, each representing distinct aspects of relationships or attributes. For example, layers could represent the different transportation options [10, 11]. As depicted in Figure 1, certain locations may be connected through trains or buses or by flight, with cross-layer connections representing the exchange options in stations and airports. For the analysis of social media platforms, the layers could be used to represent different social network platforms [12], where the same users may show different relations (e.g., friendships) on different platforms, and where cross-platform interactions can occur when content produced on a given platform is shared on a different one. In biology, it is useful to separate inhibition or catalyst interactions with different layers, but we need to track the same proteins across many different interaction types [13, 14]. When analyzing financial purchase behaviours, multilayer networks can be used, for example, to differentiate transactions made through different payment methods [15]. Indeed, the multilayer graph model has been used in different domains and applications, illustrating its large versatility. Multilayer graphs are designed to provide a more complete representation of the different and heterogeneous relationships that may characterize an entity in the graph-structured system, using the rich data available from complex systems [16] thus, providing an informative model for the underlying downstream task .

However, collecting a wide set of different relationships among a large set of entities can easily result in a significant amount of noise (e.g., incomplete, imprecise, or redundant in-

*Corresponding author

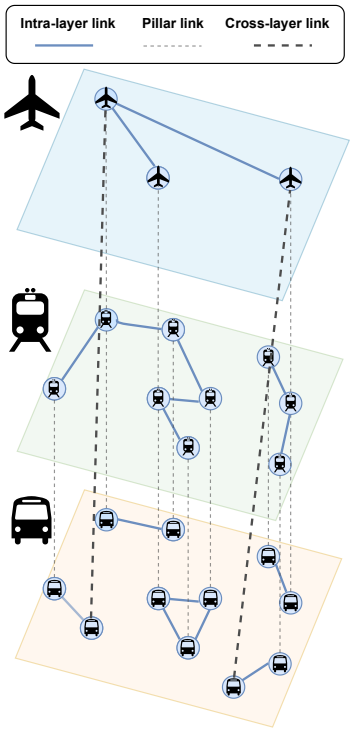


Figure 1: Example: a multilayer graph that models a transport system. Layers represent different transportation modes, from the bus on the bottom layer (orange) to the trains in the middle layer (green) and flight connections in the top layer (blue). In each layer, intra-layer connections represent the connections among airports or stations. Stations and airports usually allow exchanges among different transportation modes. Pillar links (dashed) represent stations and airports located together, a common occurrence with central stations and airport stations. Cross-layer links (dashed and bold) allow us to represent direct connections among airports or stations. In this example, two direct connections between the most remote bus stations to the airports are available.

formation) caused by the choice regarding which entities and relations should be included in the data. Single-layer graphs are already affected by this phenomenon , known as the boundary specification problem [17, 18], which is exacerbated in multilayer graphs [19]. For the case of multilayer graphs, the problem not only requires choosing which entities should be included in the graph (*horizontal boundary*) but there is also the problem of selecting which types of relations have to be included in the network, i.e., the number of layers and their semantics (*vertical boundary specification problem* [20]). While the increasing amount of information opens new research avenues [21], it can also include irrelevant knowledge related to the task at hand [16]. Therefore, it becomes crucial to conduct effective graph simplification [16], i.e., removing uninformative or redundant parts of the graph, such as entities, edges, or even layers to facilitate the final analysis .

While several machine learning techniques for the simplification of single-layer graphs have already been proposed in the literature [22, 23], for the multilayer graphs scenario only a few preprocessing heuristics, mainly unsupervised, exist [16], while cutting-edge techniques such as graph neural networks have not yet been exploited. Furthermore, work on multilayer graph neural networks [24, 25] demonstrated how crucial is to design approaches specially tailored for these complex structures, i.e., to obtain representations (embeddings) that convey the rich information present in the input graph. As a matter of fact, the straightforward application of single-layer approaches to multilayer graphs is not trivial: while a single-layer approach could be applied on each layer separately, the important interplay among the various layers would be lost. The same holds for the simplification task at the heart of this work: a framework able to thoroughly leverage the multilayer structure is of paramount importance to obtain a simplified multilayer graph properly optimized for the related downstream task.

In this work, we propose the MultilAyer gRaph simplificAtion (MARA) framework, a GNN-based framework designed to simplify multilayer graphs based on the downstream task. MARA generates node embeddings for a specific task by training end-to-end two main components: i) an edge simplification module and ii) a (multilayer) graph neural network. We tested MARA under node classification on real-world multilayer graphs from different domains. Experimental results show the effectiveness of the proposed approach: MARA dramatically reduces the dimension of the input graph not only maintaining the initial classification performances but even improving them. With MARA, we do not only enable simplification approaches that leverage single-layer simplification techniques on multilayer graphs but we also extend existing methods to work directly on multilayer graphs. Thus, with MARA, we can select the most appropriate simplification approach depending on the downstream task. Moreover, we observe that MARA can influence and enhance important graph properties, such as label assortativity. Indeed, as the selection of task-irrelevant edges is refined during the training, MARA is guided in the selection of the most important properties to preserve or enhance.

Due to the wide range of data that can be modelled as a multilayer graph, the proposed framework can have a large room of applications covering different fields like biology, physics, and health/medical analysis, where increased robustness is needed to address noise from data acquisition. Furthermore, data quality, computational performances and information visualization are also crucial aspects of any process dealing with massive amounts of graph-structured data, such as social media , communication, biological, transportation and financial systems.

2. Background

In this section, we provide background knowledge regarding the formal definition of the multilayer graph model adopted in this paper, the use of graph neural networks for the analysis of multilayer graphs, and graph simplification approaches based on deep learning. All the notations used in this paper are summarized in Table 1.

*2.1. **Multilayer graph model.***

In this subsection, we will define the main concepts for the multilayer graph model, with the definition and notations we need to formally define the framework. Since in this work, we will use ML-GCN (Multilayer Graph Convolutional Neural

Network) to instantiate MARA, for ease of reference we adopt similar definitions of multilayer graph as in the work where it was originally proposed [25]:

Definition 2.1 (Multilayer graph)**.** Given a set $\mathcal{V}$ of entities, and a set of layers $\mathcal{L} = \{L_1, ...L_l\}$ with $| \mathcal{L} | = L >= 2$, a multilayer graph is $\mathcal{G}_\mathcal{L} = (\mathcal{V}_\mathcal{L}, \mathcal{E}_\mathcal{L}, \mathcal{V}, \mathcal{L})$, where $\mathcal{V}_\mathcal{L} \subseteq \mathcal{V} \times \mathcal{L}$ is the set of entity-layer pairings or nodes (i.e., to denote which entities are present in which layers), and $\mathcal{E}_\mathcal{L} = \mathcal{V}_\mathcal{L} \times \mathcal{V}_\mathcal{L}$ is the set of directed edges between nodes.

The presence of layers implies that edges can connect nodes within the same layer or across layers. We define as *within-layer edges* the links connecting nodes in the same graph layer. Formally a link is a within-layer link when $((i,l),(j.m)) \mid (i,l),(j.m) \in \mathcal{E}_\mathcal{L}, l = m$. The within-layer edges involving a node (i,l) determine the within-layer neighborhood $\Gamma(i,l)$:

Definition 2.2 (Within-layer neighborhood)**.**

$$\Gamma(i,l) = \{(j,l) \in \mathcal{V}_\mathcal{L} | ((j,l),(i,l)) \in \mathcal{E}_\mathcal{L}\} \tag{1}$$

Within-layer edges are usually described by a set of *adjacency matrices* $A = \{A_1, ..., A_\ell\}$, where each matrix A_ℓ describes the links in the corresponding layer ℓ. These adjacency matrices describe layer-by-layer connections. However, one of the most interesting features of the multilayer graph model, is the presence of links connecting nodes in different layers, the *cross-layer edges/links*. More formally, an edge $((i,l),(j.m))$ is a cross-layer edge when $((i,l),(j.m)) \mid (i,l),(j.m) \in \mathcal{E}_\mathcal{L}, l \neq m$. In this case, when the focus is on the cross-layer links involving (i,l), we consider the *outside-layer neighborhood*. $\Psi(i,l)$ that includes all nodes reachable with cross-layer links from (i,l). The *outside-layer neighborhood* $\Psi(i,l)$ can be formally defined as:

Definition 2.3 (Outside-layer neighborhood)**.**

$$\Psi(i,l) = \{(j,m) \in \mathcal{V}_\mathcal{L} | ((j,m),(i,l)) \in \mathcal{E}_\mathcal{L}, m \neq l\} \tag{2}$$

To represent both within and cross-layer edges, we can define a Supra adjacency matrix:

Definition 2.4 (Supra adjacency matrix)**.** The supra adjacency matrix A^{sup} is:

$$A^{sup} = \begin{cases} A_l \text{ if diagonal block} \\ A_{l,m} \text{ otherwise (i.e., off the diagonal block).} \end{cases} \tag{3}$$

where $A_{l,m}$ is an adjacency matrix built upon the cross-layer connections between layer l and layer m (i.e., 1 if there exists an edge between (i,l) and (u,m) with $l = m$, and 0 otherwise).

2.2. *Graph neural networks.*

In the field of deep learning for graph-structured data, graph neural networks (GNNs) have emerged as the state-of-the-art approach in many different tasks, such as node classification [26], link prediction [27], community detection [28] and graph classification[29]. GNNs redefine basic deep learning operations, such as convolution, for graph-structured data. Thanks to their ability to make predictions leveraging the graph structure jointly with node and edge-level features, they benefit several fields such as recommender systems [30], social networks analysis [31], and network medicine [32]. In the Graph Convolutional Network (GCN) model proposed by [33], the operation of convolution on graphs is performed through an aggregation of the values of each node's features along with its

Table 1: Summary of notations used in the paper and their description.

Notations	Description
$\mathcal{G}_\mathcal{L}$	Multilayer graph
$\mathcal{V}$	Set of N entities (e.g., users)
$\mathcal{L}, \ell, L_l$	Set of layers, number of layers, l-th layer
$\mathcal{V}_\mathcal{L}$	Set of nodes in $\mathcal{G}_\mathcal{L}$
$\mathcal{E}_\mathcal{L}$	Set of edges $\mathcal{G}_\mathcal{L}$
A, A_ℓ	Adjacency matrix in G, Adjacency matrix of the l-th layer of $\mathcal{G}_\mathcal{L}$
A^{sup}	Supra-adjacency matrix
$\widetilde{A}, \widetilde{A}^{sup}$	Adjacency matrix and supra-adjacency matrix with self loops
v_i, i	Index i of a node $V_i \in \mathcal{V}_\mathcal{L}$
$\Gamma(i)$	Neighborhood of node V_i
$\Gamma(i,l)$	Within-layer neighborhood of node V_i
$\Psi(i,l)$	Outside-layer neighborhood of node V_i
X, X_l	Attribute (input feature) matrix, resp. in the l-th layer of $\mathcal{G}_\mathcal{L}$
$x, x_{(i,l)}$	Attribute (input feature) vector for node v_i, resp. node v_i in the l-th layer of $\mathcal{G}_\mathcal{L}$
f	Number of attributes (input features)
E	Edge attribute matrix
f_E	Number of edge attributes
$\mathcal{G}_{(\mathcal{L},\mathcal{X},\mathcal{E})}$	Attributed multilayer graph
d	Size of the embedding
Z, Z_l	Embedding (output feature) matrix, resp. in the l-th layer of $\mathcal{G}_\mathcal{L}$
$z_i, z_{(i,l)}$	Embedding (output feature) vector for node v_i, resp. node v_i in the l-th layer of $\mathcal{G}_\mathcal{L}$
W, W^k	Weight matrix of a generic, resp. weights of l-th GNN layers
$f_W, f_W^{(k)}$	GNN module, GNN at the k-th GNN layers
K, k	Number of GNN layers, index of a layer of the GNN
$H^{(k+1)} = f_W^{(k)}(H^{(k)}, A)$	A GNN layer computation
$f_{\theta_S}, f_{\theta_S}^k$	simplification neural network and its parameters, resp. simplification neural network for a certain GNN layer
h_i	Hidden layer vector for node v_i
$h_{(i,l)}^{(k)}$	Hidden layer vector at the k-th layer of the GNN for entity v_i in layer L_l of $\mathcal{G}_\mathcal{L}$
$Y, \hat{Y}$	Ground truth, predictions

neighbors' features. In general, deep learning models use operations like convolution to learn low dimensional latent representations for each node or edge or even entire graphs, the so-called *node/edge/graph embeddings*. If these embeddings are d dimensional, then we can expect them to be low dimensional i.e. $d << |V|$, and that similar nodes in terms of network structure will be characterized by a similar embedding. Given V a set of vertexes, X the node feature matrix, and A the adjacency matrix, a graph can be represented as $G = (V, A, X)$. A GCN model wants to compute the best possible embedding h of a node i. We do so through an aggregation process over a series of graph convolution layers, where the embeddings in each layer k, are used to compute the embeddings in the following layer $k + 1$. The aggregation in GCN involves the embeddings of the nodes i neighborhood $N(i)$, to perform the following computation:

$$h_i^{(k+1)} = \sigma\left(\sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} h_j^{(k)} W^{(k+1)}\right) \tag{4}$$

where $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ corresponds to the degree of i, computed on A_{ij} the adjacency matrix with self-loops added; W are the aggregation weights of the GCN module. The aggregation that generates the embedding $h_i^{(k+1)}$ is order-invariant, like the average function in Eq. 4.

Starting from this model, we have seen the surge of many architectures, to cover different tasks and types of graph data such as signed graphs, temporal graphs, and more recently multilayer graphs.

2.3. *Graph neural networks for multilayer graphs.*

Similarly to single layer graphs, the multilayer network embedding problem consists of learning low dimensional latent representations for each node (identified by an entity-layer pair), such that nodes that are similar in $\mathcal{G}_\mathcal{L}$ have embeddings close to each other [25]. Deep learning tasks are more challenging to apply on multilayer graphs because of the presence of intra-layer and cross-layer (also found as inter-layer) relations, different layer characteristics, as well as node features [25]. There have been some attempts to design methods and frameworks for deep learning for multilayer graphs.

MANE [34], integrates cross-layer edges for embedded representation learning, and formulates node embedding computation as an optimization problem, incorporating both intra-layer and cross-layer connections. However, it does not account for node attributes in the process. In contrast, MGCN [35] extends the GCN model to multilayer networks by constructing a GCN for each layer, using links only between nodes of the same layer and combining them in a subsequent step. The ML-GCN method [25] distinguishes itself by integrating cross-layer edges into the GCN propagation rules, enabling a more effective consideration of interlayer connections compared to MGCN. Additionally, this approach has the ability to leverage node features that are not captured by MANE . The ML-GCN framework reformulates the propagation rule of the GNN component (i.e. GCN) to aggregate topological neighborhood information from different layers. While in GCN, aggregation involves a node's features and its neighbors' features, in the ML-GCN the aggregation is performed with both its neighbors in that layer (the within-layer neighborhood) and on its neighbors located in other layers where the entity occurs (the outside-layer neighborhood). More formally:

$$h_{(i,l)}^{(k+1)} = \sigma\left(\sum_{(j,m) \in \Gamma(i,l) \cup \Psi(i,l)} \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} h_{(j,m)}^{(k)} W^{(k+1)}\right) \tag{5}$$

where $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}^{sup}$ where A_{ij}^{sup} is the supra-adjacency matrix with self-loops added.

2.4. *Deep learning for graph simplification.*

Graph simplification consists of removing uninformative or redundant parts of the graph while keeping almost all information of the input graph [36]. While there are many works on simplification [37], only a few are focused on simplification for deep learning on graphs. DropEdge [36] simplifies the graph for a GNN model (e.g. GCN) by randomly removing a fraction of the edges from the input graph during the training phase. The method influences only the training phase, while during validation and testing the removal is not performed. The evaluation of DropEdge shows that even a random removal can lead to similar or improved performance across different tasks, such as node classification and link prediction. As noted in [36], even when performance gain is not significant, the advantage of simplification lies in the fact that the randomness and the diversity of the input graph are increased, thus reducing the risk of overfitting (i.e., when the gap between the training error and test error is too large because the model learns properties of the training set that are not present in the test set [38]). Moreover, removing edges makes links more sparse, which helps to reduce the impact of *over-smoothing*, i.e., the phenomena that occurs when the node-specific information is lost after several iterations of GNN message passing [39], leading to very similar embeddings for every node.

However, this approach has a key limitation: only the graph neural network component is trained, while the simplification module cannot improve during training. Therefore, some approaches were introduced, that rely on a deep learning based simplification module whose parameters can be tuned during training. In this case, the approaches train both components end to end: this is the case of NeuralSparse, presented in [22]. In NeuralSparse [22], the simplification process is done through the deep neural network: during the training phase, the deep neural network learns a simplification strategy that favors downstream tasks. In the testing phase, the neural network is used to select the edges to remove from the input graph, based on the learned strategy. The neural network model, i.e., the multilayer perception (MLP), is given in input an edge's features and the features of the nodes it connects and uses them to compute a score, that will be higher when the method thinks it is worth it to keep a certain edge in the graph. During training, the selection process revolves around a sampling procedure, revolving around *k-neighbor subgraphs*. In practice, according to a hyperparameter k, the method will select for each

node only a subset k of its neighbors. The selection is done for each node, leading to a new simplified graph, that is used for convolution by the GNN. The key advantage is that, in this case, the simplification module has parameters that can be adjusted during training. However this is not straightforward, since the sampling process is stochastic and thus essentially non-differentiable. In NeuralSparse, the solution consists of performing the sampling process with a function that can approximate the sampling from a categorical distribution, in our case the selection among edges, but can be made differentiable through some mathematical reformulations. They selected the Gumbel-Softmax [40, 41], a method that can approximate the sampling from a categorical distribution and most importantly, it is differentiable using a reparameterization trick [41]. The reparameterization trick rewrites the stochastic sampling process as a linear combination of two components, one deterministic and the other stochastic [42]. The deterministic part can be adjusted through backpropagation, while the stochastic part can be ignored. Therefore, the simplification module is now trained during the process. Instead, during validation and testing, the graph can be simplified relying on the neural network component, without the reliance on sampling. When features are informative for the task, the selection process should be more accurate, thus leading to a more precise selection process and potentially to improvements in terms of performance.

Other works rely on similar principles. In AdaptiveGCN [43] simplification process is led by a deep neural network like in NeuralSparse, but a simplification step is performed before each graph convolution step. In PTDnet [44] additional constraints on the simplification process are introduced, encouraging the removal of more edges or prioritizing the simplification of edges connecting different node clusters. Other works such as [45] and [23] have designed frameworks for simplification with reinforcement learning. Note that while there are several works on single-layer graph simplification, there is a lack of work relying on deep learning for the simplification of multilayer graphs.

3. Research questions

From the literature, it becomes clear that graph simplification has many advantages, such as the limitation of overfitting, that can lead to better generalization performances and it also limits the effects of over-smoothing, thus allowing for deeper models [36]. But while there are several works on single-layer graph simplification, there is a lack of works relying on deep learning for the simplification of multilayer graphs, mainly because the applications of single-layer methods in the multilayer case are not straightforward. Given the benefits of graph simplification and the usefulness of the multilayer graph model, it is very important to fill this research gap. Therefore, in this work, we face the problem of understanding how we can apply the current deep learning based approaches designed for single-layer graphs to multilayer graphs. Among various aspects, we would like to see how graph simplification methods influence prediction performances, compared to single-layer cases. Moreover, we would like to deepen our understanding of the simplification process, especially when the methods can tune their selection strategy. These aspects can be summarized in the following research questions:

Research question RQ1: What is the impact of graph simplification performed on multilayer graphs?

Research question RQ2: How does graph simplification influence the structure of multilayer graphs?

Research question RQ3: How is prediction performance affected by the graph simplification hyperparameters?

4. The MARA framework

In order to address our research questions, in this work, we introduce a framework for the simplification of multilayer graphs and evaluate the impact of a simplification approach on a machine learning task. We evaluate the impact of graph simplification approaches on a typical learning task, i.e., node classification. In this section, we formally present the problem and the framework.

*4.1. **Problem definition.***

The graph simplification problem on single-layer graphs can be defined as follows: given a graph $G(V, E, X_E, X_V)$, where V is a set of n nodes, $E \subset V \times V$ is the set of edges; X_V is a set of node attributes, X_E is a set of edge attributes. Simplification tries to obtain a subgraph of G, that would be $G' = G(V', E', X_E, X_V)$, where $V' \subset V \vee E' \subset E$ i.e the number of nodes and/or edges is reduced. Similarly, on a multilayer graph, simplification can be defined as the problem of obtaining a graph $f_{\theta_S}(\mathcal{G}_\mathcal{L}) = \mathcal{G}_\mathcal{L}{}' = (\mathcal{V}_\mathcal{L}{}', \mathcal{E}_\mathcal{L}{}', \mathcal{V}', \mathcal{L}')$ so that the number of nodes and/or edges is reduced. Formally, we are looking for a simplified multilayer graph $\mathcal{G}_\mathcal{L}{}'$. such that the following disjunction of conditions holds: $| \mathcal{V} | < | \mathcal{V}' | \vee | \mathcal{L} | < | \mathcal{L}' | \vee | \mathcal{V}_\mathcal{L} | < | \mathcal{V}_\mathcal{L}{}' | \vee | \mathcal{E}_\mathcal{L} | < | \mathcal{E}_\mathcal{L}{}' |$. In the following, we present the framework to compute the simplified multilayer graph.

*4.2. **The simplification framework.***

We propose the MultilAyer gRaph simplification (MARA) framework, a GNN-based approach designed to simplify multilayer graphs based on the downstream task. An overview of the framework is presented in Figure 2.

MARA generates node embeddings for a specific task by training jointly two main components: i) an edge simplification module and ii) a (multilayer) graph neural network. Based on this framework, we propose two approaches to perform graph simplification on a multilayer graph: i) Layer by layer graph simplification and ii) Multilayer graph simplification. We now present the two concepts behind them.

Layer by layer graph simplification. To perform graph simplification on a multilayer graph by exploiting methods for single-layer graphs, we can use a layer-by-layer approach. In the layer-by-layer simplification, methods are applied to each layer before recomposing the supra-adjacency matrix: cross-layer links are not involved. We can define a layer graph as $G[\ell]$

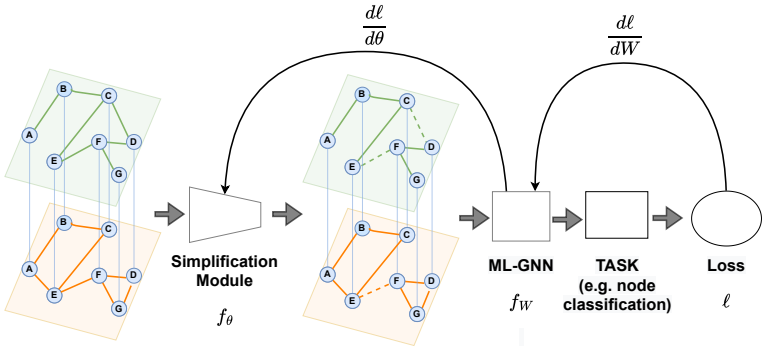
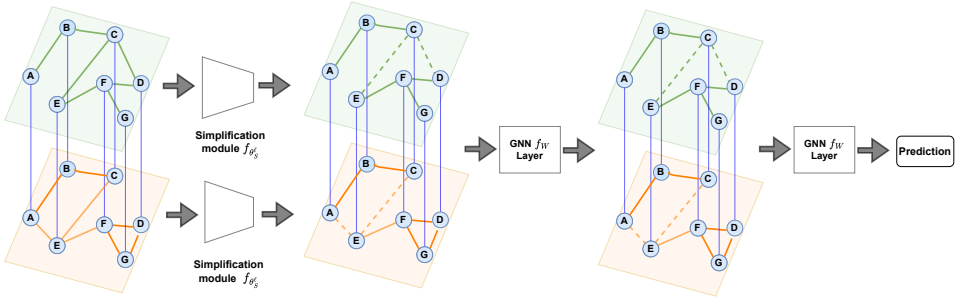
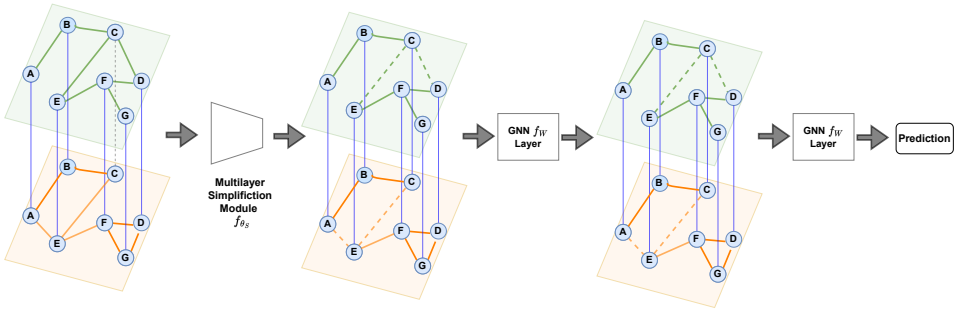


Figure 2: **Proposed multilayer graph simplification framework.** Overview of the proposed framework. A simplification module f_θ and multilayer graph neural network f_W, are used to generate node embeddings for a downstream task. If the simplification module is trainable e.g. a neural network, it is possible to train the two components jointly: through gradient descent, we update the parameters θ, W backpropagating from the loss function ℓ. In this case, the simplification module can learn to detect noisy links specifically for the downstream task.



(a) **Layer by layer graph simplification with multilayer GNN**. A simplification module (simplification neural network $f_{\theta_S^\ell}$) detects the links to remove at each layer ℓ of the input multilayer graph, while a GNN (multilayer graph neural network f_W) generates embeddings for a downstream task.



(b) **Multilayer graph simplification with multilayer GNN.** A multilayer simplification module (simplification neural network f_θ) detects the links to remove by taking into account the whole input multilayer graph, while a GNN (multilayer graph neural network f_W) is used to generate node embeddings for a downstream task.

Figure 3: Overview of the proposed approaches for multilayer graph simplification: (a) *layer-by-layer* and (b) *multilayer*. Note that the difference between the two approaches lies in the simplification process, while the use of the GNN is the same.

where every edge connects nodes in the same layer ℓ. Therefore, at each layer ℓ a simplification neural network $f_{\theta_S^\ell}$ detects noisy links over the layer-graph $G[\ell]$, generating a new version of the graph that we can define as $G[\ell]'$. The simplified graphs are used to update A'^{sup}, which will be used to train the graph neural network.

It is important to note that simplification can be applied at a different *stage*s of the process: we can simplify *once* or before *each* graph convolutional layer. In the first case, a simplification module detects noisy elements while a graph neural network model is used to generate node embeddings for a downstream task. Here, simplification occurs only *once*, so that the graph is the same at each GNN layer. In the other case, at each GNN layer, a simplification module detects noisy elements while a graph neural network model generates node embeddings for a downstream task. The simplification is performed multiple times so that before *each* GNN layer, we are considering different versions of the graph.

MARA allows training with both simplification stages. The training phase for layer-by-layer graph simplification is summarized in Algorithm 1.

Multilayer graph simplification. To define a simplification

Algorithm 1 Training step for layer-by-layer simplification with ML GNN

Input: training multilayer graph $\mathcal{G}_{\mathcal{L}}$, graph neural network f_W, simplification neural network f_{θ_S}, simplification stage *stage*.
Output: Embeddings for downstream task
function: LayerByLayerSimplificationTrainingStep($\mathcal{G}_{\mathcal{L}}$,$f_W$, f_{θ_S},*stage*)
$A^{sup} \leftarrow \mathcal{G}_{\mathcal{L}}$.getSupraAdjacencyMatrix()
$H^0 \leftarrow \mathcal{G}_{\mathcal{L}}.X_v$ ▷ number of GNN hidden layers
$\mathcal{L} \leftarrow \mathcal{G}_{\mathcal{L}}.X_v$ ▷ number of graph layers
$K = f_W.K$ ▷ number of layers in the graph
if *stage* = "once" **then** ▷ Simplify graph just once
    temp = array()
    for layer $\ell \in 1...\mathcal{L}$ **do**
        $G \leftarrow \mathcal{G}_{\mathcal{L}}$.getLayerSubgraph($l$)
        $A_\ell \leftarrow G$.getAdjacencyMatrix()
        $A'_\ell \leftarrow f_{\theta_S^\ell}(A_\ell, G.X_V, G.X_E)$ ▷ simplify layer subgraph G
        temp.append(A'_ℓ)
    end for
    $A'^{sup} \leftarrow$ Merge *temp* into a supra adjacency matrix
end if
for $k = 1...K$ **do** ▷ GNN layers activation
    if *stage* = "each" **then** ▷ Different graph every time
        temp = array()
        for layer $\ell \in 1...\mathcal{L}$ **do**
            $G \leftarrow \mathcal{G}_{\mathcal{L}}$.getLayerSubgraph($l$)
            $A_\ell \leftarrow G$.getAdjacencyMatrix()
            $A'_\ell \leftarrow f_{\theta_S^\ell}(A_\ell, G.X_V, G.X_E)$ ▷ simplify layer subgraph G
            temp.append(A'_ℓ)
        end for
        $A'^{sup} \leftarrow$ Merge *temp* into a supra adjacency matrix
    end if
    $H^k \leftarrow f_W^{(k-1)}(H^{(k-1)}, A'^{sup})$ ▷ hidden representations update
end for
Backpropagation to update f_{θ_S}, f_{θ_W}
return trained f_{θ_S}, trained f_{θ_W}

methodology conceived explicitly for a multilayer graph, able to properly take into account the complex structure of such a model, we propose to use a simplification neural network f_θ that detects noisy edges and a graph neural network f_W to generate node embeddings for a downstream task (cf. Fig. 3b). The key difference with respect to the single-layer counterpart is that the simplification module is unique, and acts directly on the supra-adjacency matrix A^{sup} to generate the simplified A'^{sup}. Acting directly on the supra-adjacency matrix also has an additional advantage: the simplification module can remove noisy or redundant cross-layer links as well. Even in the multilayer simplification case, simplification can be applied at different *stage*s: we can simplify *once* (i.e., the graph is the same at each GNN layer) or before *each* graph convolutional layer (i.e., the simplification is performed multiple times, so that each GNN layer works on a different version of the graph). The training phase for multilayer graph simplification is summarized in Algorithm 2.

Algorithm 2 Training step for multilayer simplification with ML GNN

Input: training multilayer graph $\mathcal{G}_{\mathcal{L}}$, graph neural network f_W, simplification neural network f_{θ_S}, simplification stage *stage*.
Output: Embeddings for downstream task
function: MultilayerSimplificationTrainingStep($\mathcal{G}_{\mathcal{L}}$,$f_W$, f_{θ_S},*stage*)
$A^{sup} = \mathcal{G}_{\mathcal{L}}$.getSupraAdjacencyMatrix()
$H^0 = \mathcal{G}_{\mathcal{L}}.X_v$ ▷ Initial embeddings are node features
$K = f_W.K$ ▷ number of GNN hidden layers
if *stage*= "once" **then** ▷ Simplify graph just once
    $A'^{sup} \leftarrow f_{\theta_S}(A^{sup}, \mathcal{G}_{\mathcal{L}}.X_V, \mathcal{G}_{\mathcal{L}}.X_E)$ ▷ Simplify
end if
for $k = 1...K$ **do** ▷ GNN layers activation
    if *stage* = "each" **then** ▷ Different graph every time
        $A'^{sup} \leftarrow f_{\theta_S}(A^{sup}, \mathcal{G}_{\mathcal{L}}.X_V, \mathcal{G}_{\mathcal{L}}.X_E)$ ▷ Simplify
    end if
    $H^k \leftarrow f_W^{(k-1)}(H^{(k-1)}, A'^{sup})$ ▷ hidden representations update
end for
Backpropagation to update f_{θ_S}, f_{θ_W}
return trained f_{θ_S}, trained f_{θ_W}

5. Experimental evaluation

In this section, we present the dataset we collected or generated and the experimental setting we used to perform the framework evaluation.

5.1. ***Data.***

For the experimental evaluation, we selected datasets from different domains showing different structural characteristics. Dataset characteristics are summarized in Table 2.

Table 2: Summary of structural characteristics of the graph datasets: type of the graph, number of layers (L), number of nodes ($|V|$), number of edges ($|E|$), density (mean/SD) over the layers (d), and number of classes (C)

dataset	L	$\|V\|$	$\|E\|$	d	C
imdb-mlh	2	5614	23208	0.0007 ± 0.0000	3
um-econ	2	15414	224855	0.0018 ± 0.0012	4
um-socioeco	4	18212	1199863	0.0138 ± 0.0118	4
Koumbia 2	2	4492	18783	0.0010 ± 0.0001	2
Koumbia 5	5	11230	91938	0.0010 ± 0.0002	2

All the selected multilayer graph datasets are associated with real-word node features, a characteristic that can be leveraged by a simplification module to guide the underlying decision process. The ***um-econ*** and ***um-socioeco*** [12] multilayer graphs

Table 3: AUC (mean and standard deviation over 3 random seeds [46]) obtained by the baseline and MARA.

model	simp	data stage	um-econ	um-socioeco	imdb-mlh	Koumbia 2	Koumbia 5
GNN	-	-	0.7420 ± 0.0022	0.6939 ± 0.0234	0.8035 ± 0.0218	0.9056 ± 0.0049	0.9237 ± 0.0033
MARA (DE)	multi	once	0.7451 ± 0.0128	0.6936 ± 0.0279	**0.8135 ± 0.0351**	0.9068 ± 0.0007	0.9228 ± 0.0041
		each	0.7487 ± 0.0150	0.6905 ± 0.0233	0.8122 ± 0.0324	0.9042 ± 0.0075	0.9246 ± 0.0069
	l-b-l	once	0.7407 ± 0.0083	0.6939 ± 0.0234	0.8005 ± 0.0253	0.9059 ± 0.0059	0.9252 ± 0.0063
		each	0.7418 ± 0.0102	0.6988 ± 0.0130	0.8079 ± 0.0280	0.9022 ± 0.0045	0.9238 ± 0.0051
MARA (NS)	multi	once	**0.7522 ± 0.0084**	0.6924 ± 0.0208	0.8011 ± 0.0299	0.9023 ± 0.0042	0.9223 ± 0.0138
		each	0.7458 ± 0.0107	0.6817 ± 0.0347	0.7987 ± 0.0257	0.9080 ± 0.0023	0.9244 ± 0.0093
	l-b-l	once	0.7438 ± 0.0113	**0.7199 ± 0.0099**	0.8077 ± 0.0260	0.9087 ± 0.0045	0.9205 ± 0.0022
		each	0.7457 ± 0.0008	0.7076 ± 0.0423	0.8046 ± 0.0249	**0.9103 ± 0.0052**	**0.9281 ± 0.0067**

describe user interactions in a decentralized social media platform (Steemit) [47, 48]. In these graphs nodes are users, and layers are interactions of different types. Users can engage in various actions that form connections between them, either explicitly or implicitly. Key operations include social activities typical of traditional social networks, such as posting, rating, voting, sharing, and following. For instance, "following" is an explicit action connecting two users, while voting for a post or comment by another user forms an implicit relationship. Additionally, users can participate in monetary operations involving the trading of cryptocurrency tokens, which the platform distributes to incentivize participation (a detailed recap can be found here [47]). In our framework, different layers separate various types of interactions, with each intra-layer connection representing a different relationship type. For example, the "following" relationship layer is distinct from the trading relationship layer. Cross-layer links connect the same user across these layers. Although the platform supports numerous interactions, our focus is on the most common and significant ones. User labels describe their migration to a different social media platform, called Hive (4 cases: inactive, stay, leave, active on both as defined in previous works [49, 50]). User migration has gained increasing relevance [51, 52, 53, 54], particularly with the emergence of new-generation platforms [55, 56, 57, 50, 12]. Although the Steemit platform supports numerous interactions, our focus is on the most common and significant ones, as in previous works. The *um-econ* dataset is a subgraph composed of 2 layers of economic interactions, while *um-socioeco* considers interaction on 4 layers, 2 social and 2 economic.In addition, to graph structure, we also have user features derived from graph-based metrics, such as PageRank [58]. ***IMDb-mlh*** [59] is a multilayer graph derived from IMDb. IMDb (Internet Movie Database) is an online database offering detailed information about films, TV shows, video games, and streaming content, including cast, crew, plot summaries, and user reviews. For its richness in terms of data, it has been used as a data source for many machine learning tasks and network analysis tasks [60, 61, 62, 63]. Different multilayer graphs can be derived from this data source, depending on the selected task, in this work, we rely on the version used in [59], where nodes are movies and two movies are connected if they share either an actor or a director. The layers separate the type of connection, so the intra-layer layer can represent either a shared actor or a shared director. Cross-layer links, similar to the previous case, maintain the identity, linking the same movies across the layers. Movie labels describe the movie type (action, comedy, drama). In addition, node features are generated from a text summary of the plot, to be leveraged by the machine learning models. Finally ***Koumbia 2*** and ***Koumbia 5*** [64, 25] are multilayer graphs extracted from a time series of Sentinel-2[1] optical satellite images, covering the agricultural landscape of Koumbia in Burkina Faso. The graphs are generated via the geo2net framework[2], which allows to derive multilayer graphs from satellite images with an arbitrary number of layers, for a customizable detailed representation of the dataset. In the derived graphs, the nodes represent segments of the satellite image, and labels correspond to either crop (cultivated areas) or no-crop (uncultivated areas, such as forests) segments. Layers correspond to functional classes (e.g., temporal radiometric profiles).More precisely, the framework infers both nodes and edges from Satellite Image Time Series, and similarly layer memberships and edge weights are also derived this way, for both intra-layer links and cross-layer links. A key difference from other datasets is that cross-layer links can link both the same nodes across layers but can also link different nodes across different layers, with different weights determined by the geo2net framework. The network also includes real-world attributes for each node, corresponding to a time series of radiometric statistics for each segment.

*5.2. **Experimental setting.***

In this work, we focus on node classification tasks, i.e., we learn the embeddings required to predict the label associated with each node in the graph. As GNN for MARA we select the GCN, but note that any other GNN model can be leveraged. As a baseline, we consider the multilayer GNN [25] (***GNN***). As previously discussed, MARA can be equipped with different simplification strategies as well. In this work, we selected i) DropEdge [36] (MARA(DE)), the single-layer graph simplification method that randomly removes edges with probability p, and ii) NeuralSparse [22] (MARA(NS)), which is

[1] https://sentinel.esa.int/web/sentinel/missions/sentinel-2

[2] https://gitlab.irstea.fr/raffaele.gaetano/geo2net

able to leverage node features to select a subset of edges to keep (a subgraph-based selection process is performed where for each node only k of its neighbors are kept as well as their connecting edges). Note that both approaches were originally designed for single-layer simplification, hence for this work, we extended them to perform multilayer (*multi*) and layer-by-layer (*l-b-l*) simplification (cf. Section 4). Moreover, each implementation can be applied at different *stages*: we can simplify *once* or before *each* graph convolution layer (cf. Section 4). For MARA(DE), we consider different drop rate probabilities $p = \{0.1, 0.3, 0.5, 0.7\}$, while for MARA(NS), we assess different $k = \{5, 10.15\}$, with τ varying during training as in [22]. We perform all the experiments with a transductive learning setting like in [25]. In a transductive setting, all node attributes and topological information can be used for training, while only a subset of labels is visible to the GNN model. All models were trained using the Adam optimization algorithm [65] with full batch training [33], L2 weight regularization set to 0.0005. For each graph and method, the average accuracy was computed over 3 independent runs, where each run corresponded to a different train-validation-test split, with 25% of training entities as previously done in [25] and the rest split in validation (25%) and test entities (50%). The combination of hyperparameters with the best average validation metric is selected, and we report the final test metric. Due to the huge number of combinations, we rely on early stopping, training for 250 epochs with 10 epochs of patience (reloading the best model). As an evaluation metric, we select AUC (Area under the ROC Curve) as done in [22], since it is well suited for datasets showing unbalanced label distribution, such as *Imdb*, *um-econ* and *um-socioeco*.

6. Results

In this Section, we report the experimental results and the related discussion with the aim of providing answers to the aforementioned research questions (Section 3).

6.1. Framework evaluation.

We first focus on the research question *RQ1* by targeting the performance achieved by the simplification methods.

Table 3 reports the average AUC scores on the test set. We can observe how MARA generally improves upon the GNN baseline, and systematically achieves the best performances. Note that MARA(NS) almost consistently outperforms MARA(DE), demonstrating the importance of exploiting node features for the simplification task. The only exception is represented by *imdb-mlh*, where features information improves the performance, but the MARA(DE) variant obtains even better performance. Additional insights can be obtained by comparing the multilayer (*multi*) vs layer-by-layer (*l-b-l*) and the *once* vs *each* approaches. Regarding MARA(DE), we note that *multi* tends to be more effective on 2-layer graphs (i.e., *um-econ*, *um-socioeco* and *Koumbia-2*) while *l-b-l* seems to be more effective in presence of a greater number of layers. Note also that, with the (DE) variant, simplifying *once* tends to be the winning choice. This is consistent with the stochastic nature of this approach, i.e., repeating a random process at each layer may negatively impact the result. As concerns MARA(NS), *l-b-l* tends to be the best choice in most cases: it may be because the NeuralSparse simplification strategy is based on a single-layer notion of a node's neighborhood. Devising an advanced strategy to properly take into account the multilayer neighborhood is left as future work. In terms of when to simplify (stage), for the (NS) variant, we can see that simplifying *once* brings better results for datasets showing an unbalanced distribution of the labels (i.e., *um-econ*, *um-socioeco* and *imdb-mlh*), while simplifying before *each* convolution layer seems the best approach for the more balanced *Koumbia* graphs.

Overall, **MARA leads to significant performance improvements, while the variety of proposed approaches allows MARA to find the most suitable simplification approach for tasks of different domains.**

6.2. Analysis of simplified graphs.

In this section, we discuss how the simplification process impacts the structural characteristics of the multilayer graphs, providing an answer to research question *RQ2*. For each dataset, we compare graph structure before and after the simplification with MARA. We show results for each prediction sub-task, i.e., user migration prediction on *um-econ* (Table 4), movie classification on *imdb-mlh* (Table 5) and cropland mapping on *Koumbia-2* (Table 6)while the other results can be consulted in the Appendix. For the analysis, we first focus on the *um-econ* case (Table 4). It can be noted how the impact of MARA(NS) can be different on each layer of a specific graph, while the action of MARA(DE) seems to be more uniform over a given graph. Once again, this is consistent with the fact that one approach leverages node features while the other is a random approach. The clearest impact is observed on the number of intra-edges, **MARA drastically reduces the number of edges while still improving the performance**: this is extremely important as the computation cost of graph convolution is linear in the number of graph edges [33], making a reduced number of edges an ideal property. In addition, some interesting observations can be drawn about label assortativity, i.e., the similarity of connections in the graph with respect to node labels (high label assortativity means that a node is more likely to connect with a node with the same label). We can see how **MARA(NS) tends to increase label assortativity across layers**: this makes sense as MARA(NS) can leverage node features, so it would be able to preserve the connections between similar nodes. Such behavior cannot be replicated by the random procedure behind MARA(DE). Similarly, as regards transitivity (i.e., the fraction of all possible triangles present in a graph), we can observe a general decrease, since the number of triangles is necessarily reduced as we remove edges. However, on layers with lower transitivity (< 0.1), only MARA(NS) increases transitivity values: this can be observed in *um-econ* and *um-socioeco*.

The relevance of training jointly simplification and graph neural network is, therefore, the most important observation: during the training, MARA(NS) improves its capacity to recognize edges that are unrelated to the task at hand, allowing it to determine which graph characteristics are most crucial to

Table 4: Statistics for each graph layer before and after the simplification on um-econ dataset.

	ℓ	Intra-layer edges	Label assortativity	Transitivity	Indegree mean	Indegree max	Outdegree mean	Outdegree max
MARA (NS)	L0	174381.00	0.08	0.01	23.63	3610.00	23.63	6021.00
		6207.00	0.35	0.02	1.17	328.00	1.02	3.00
		(-96.44%)	(+320.57%)	(+86.70%)	(-95.06%)	(-90.91%)	(-95.69%)	(-99.95%)
	L1	35060.00	0.27	0.00	5.55	937.00	5.55	4769.00
		5038.00	0.62	0.02	0.87	145.00	1.02	3.00
		(-85.63%)	(+127.57%)	(+2947.23%)	(-84.39%)	(-84.53%)	(-81.71%)	(-99.94%)
MARA (DE)	L0	174381.00	0.08	0.01	23.63	3610.00	23.63	6021.00
		121999.00	0.08	0.01	16.53	2552.00	16.53	4230.00
		(-30.04%)	(+2.17%)	(-31.27%)	(-30.03%)	(-29.31%)	(-30.03%)	(-29.75%)
	L1	35060.00	0.27	0.00	5.55	937.00	5.55	4769.00
		24578.00	0.27	0.00	3.89	642.00	3.89	3349.00
		(-29.90%)	(-1.44%)	(-31.29%)	(-29.87%)	(-31.48%)	(-29.89%)	(-29.78%)

Table 5: Statistics for each graph layer before and after the simplification on imdb-mlh dataset.

	ℓ	Intra-layer edges	Label assortativity	Transitivity	Indegree mean	Indegree max	Outdegree mean	Outdegree max
MARA (NS)	L0	6121.00	0.70	0.40	4.27	79.00	4.27	79.00
		2818.00	0.87	0.29	3.09	42.00	3.09	40.00
		(-53.96%)	(+23.27%)	(-28.36%)	(-27.55%)	(-46.84%)	(-27.55%)	(-49.37%)
	L1	5355.00	0.72	0.38	4.00	69.00	4.00	69.00
		2816.00	0.90	0.00	3.09	42.00	3.09	38.00
		(-47.41%)	(+24.60%)	(-100.00%)	(-22.63%)	(-39.13%)	(-22.63%)	(-44.93%)
MARA (DE)	L0	6121.00	0.70	0.40	4.27	79.00	4.27	79.00
		4277.00	0.71	0.26	3.00	55.00	2.98	53.00
		(-30.13%)	(+1.44%)	(-34.14%)	(-29.80%)	(-30.38%)	(-30.27%)	(-32.91%)
	L1	5355.00	0.72	0.38	4.00	69.00	4.00	69.00
		3749.00	0.73	0.26	2.79	46.00	2.81	49.00
		(-29.99%)	(+0.47%)	(-29.83%)	(-30.22%)	(-33.33%)	(-29.71%)	(-28.99%)

Table 6: Statistics for each graph layer before and after the simplification on Koumbia 2 dataset.

	ℓ	Intra-layer edges	Label assortativity	Transitivity	Indeg mean	Indegree max	Outdegree mean	Outdegree max
MARA (NS)	L0	5724.00	0.72	0.16	4.39	20.00	4.39	24.00
		2254.00	0.90	0.00	2.85	11.00	2.85	11.00
		(-60.62%)	(+24.26%)	(-100.00%)	(-35.18%)	(-45.00%)	(-35.18%)	(-54.17%)
	L1	4779.00	0.79	0.20	3.97	25.00	3.97	27.00
		2253.00	0.91	0.00	2.85	22.00	2.85	20.00
		(-52.86%)	(+15.07%)	(-100.00%)	(-28.32%)	(-12.00%)	(-28.32%)	(-25.93%)
MARA (DE)	L0	5724.00	0.72	0.16	4.39	20.00	4.39	24.00
		2909.00	0.72	0.08	2.21	13.00	2.22	15.00
		(-49.18%)	(-0.67%)	(-52.59%)	(-49.64%)	(-35.00%)	(-49.55%)	(-37.50%)
	L1	4779.00	0.79	0.20	3.97	25.00	3.97	27.00
		2356.00	0.79	0.09	1.97	13.00	1.97	17.00
		(-50.70%)	(+0.24%)	(-53.31%)	(-50.41%)	(-48.00%)	(-50.50%)	(-37.04%)

maintain or enhance. Additionally, with both variants, MARA demonstrates the capability of significantly reducing the number of edges while improving or at least keeping performance.

6.3. ***Hyperparameters sensitivity analysis.***

As a last analysis step, we address RQ3, which requires us to evaluate the impact of the hyperparameters on the graph simplification methods. To this end, we study the impact of varying the main hyperparameters, i.e., the drop rate p for MARA(DE) and k for MARA(NS). In Figure 4 we report a sensitivity anal-

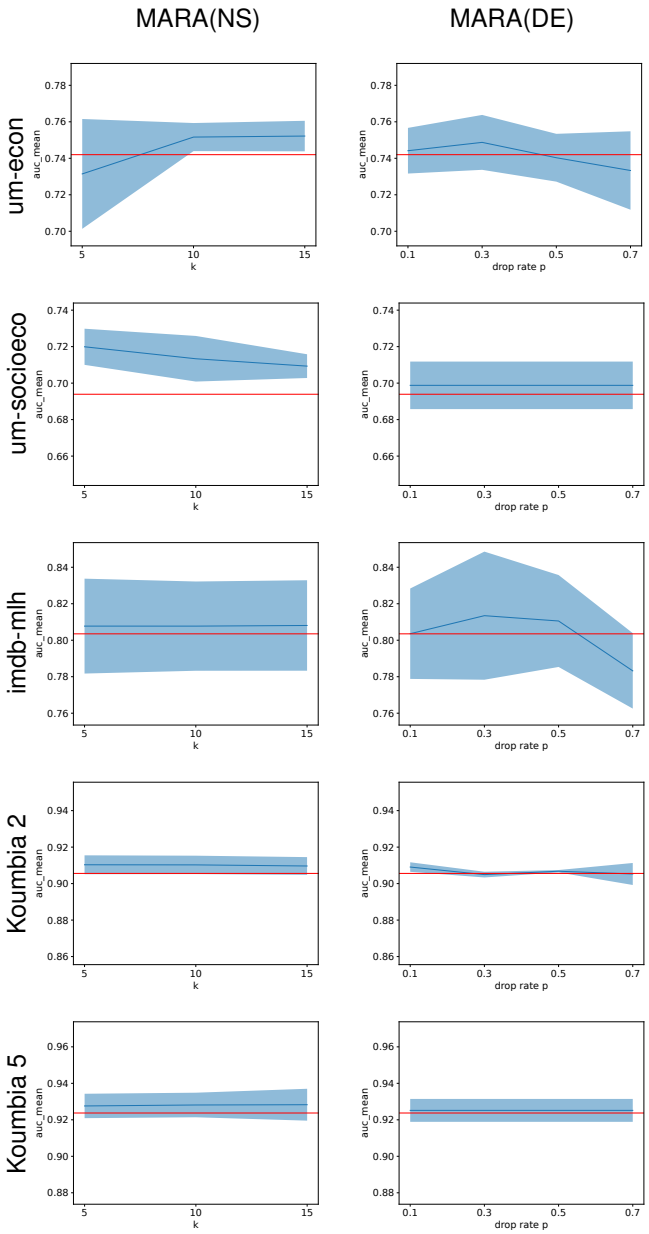


Figure 4: Sensitivity analysis based on AUC, for the 2-layer graphs. We compare the AUC for the baseline (in red), with the AUC (average, standard deviation) for the best simplification method (in blue).

ysis for p and k, where the other hyperparameters are set to the best performing combination. We can see that for *um-econ*, low k values lead to lower performance. Similarly, a high drop rate of p seems to lead to worse performance. For *imdb-mlh*, we can draw similar observations for p (i.e., a high drop worsens the performance), while variations of k seem to have a minor impact on the process. Similarly, the impact of k is minor also for *Koumbia-2*. In this case, the impact of p seems to be reduced too.

Overall, the takeaway is that **both MARA(NS) and MARA(DE) are robust to variations of their respective main hyperparameters** k and p. This makes them solid and easy to deploy, by making hyperparameter tuning relatively unimportant.

7. Conclusions

The findings presented in this work show the significance of the proposed framework: MARA leads to significant performance improvements, by selecting the best available simplification strategies. These advances in performance are even more noteworthy when we take into account that MARA achieves them while drastically reducing the number of edges. Most importantly, MARA shows the importance of jointly training the simplification of the multilayer graphs and the node classification tasks. As the ability to identify task-irrelevant edges increases, MARA is guided to discover the most important graph properties to preserve or enhance.

These findings hold practical significance with direct applications. Providing a methodology to easily perform multilayer graph simplification can help scholars and practitioners from multiple fields to improve both scalability and prediction performances in their everyday tasks. Considering interdisciplinary research environments, a simplification method can also favour the interplay between domain experts and data scientists. Domain experts would be allowed to collect and model a large number of relations, regardless of the fact that some of these data may be considered noisy or negligible for a specific task, since data scientists could, in turn, enhance the quality of a given model for a downstream task through graph simplification. In this regard, studying the graph simplification model also provides an opportunity for improved model understanding. As the model learns to select the most important graph parts, we obtain not only a simpler graph but also an opportunity to understand what entities, relations and layers are central for the outcomes of the selected model to be reliable. This method can be useful in any setting with complex and rich multilayer network structures, like the one covered by the considered datasets.

Future research will focus on analyzing how multilayer simplification can be beneficial for a variety of tasks, including link prediction (removing unimportant or "spam" links to improve prediction performance), clustering (removing redundant links should improve boundaries between clusters, thus improving cluster quality), and graph classification (removing noisy links should help in the identification of similar graphs). Finally, additional future works will focus on the interaction between graph properties and downstream tasks to support multilayer simplification. A better understanding of graph properties can be beneficial in the development of simplification algorithms and overall it could lead to a better understanding of complex systems spanning different domains.

Acknowledgements

This work has been partially funded by the Italian Ministry of University and Research (MUR) and the European Union – NextGenerationEU in the framework of the PRIN 2022 project "AWESOME: Analysis framework for WEb3 SOcial MEdia" –

CUP: I53D23003680006. This work was supported in part by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. This work is also partly supported by Cisco Systems.

CRediT authorship contribution statement

Cheick Tidiane Ba: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Roberto Interdonato**: Writing – original draft, Writing – review & editing, Validation, Methodology, Supervision, Project administration, Formal analysis. **Dino Ienco**: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Sabrina Gaito**: Conceptualization, Methodology, Resources, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

- None of the authors of this paper has a financial or personal relationship with people or organizations that may inappropriately influence or bias the content of the paper.
- It is to specifically state that "No Competing interests are at stake and there is No Conflict of Interest" with other people or organizations that could inappropriately influence or bias the content of the paper.
- The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, C. Zhou, Synchronization in complex networks, Physics reports 469 (3) (2008) 93–153.

[2] W. Thompson, P. Brantefors, P. Fransson, From static to temporal network theory: Applications to functional brain connectivity, Network Neuroscience 1 (2017) 1–56. doi:10.1162/NETN_a_00011.

[3] M. Nekovee, Y. Moreno, G. Bianconi, M. Marsili, Theory of rumour spreading in complex social networks, Physica A: Statistical Mechanics and its Applications 374 (1) (2007) 457–470.

[4] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, A. Vespignani, Epidemic processes in complex networks, Rev. Mod. Phys. 87 (2015) 925–979. doi:10.1103/RevModPhys.87.925.
URL https://link.aps.org/doi/10.1103/RevModPhys.87.925

[5] L. Zhao, G.-J. Wang, M. Wang, W. Bao, W. Li, H. E. Stanley, Stock market as temporal network, Physica A: Statistical Mechanics and its Applications 506 (2018) 1104–1112. doi:10.1016/j.physa.2018.05.039.
URL http://dx.doi.org/10.1016/j.physa.2018.05.039

[6] S. Battiston, G. Caldarelli, M. D'Errico, The financial system as a nexus of interconnected networks, in: Interconnected networks, Springer, 2016, pp. 195–229.

[7] J. L. Curzel, R. Lüders, K. V. O. Fonseca, M. Rosa, Temporal performance analysis of bus transportation using link streams, Mathematical Problems in Engineering (2019).

[8] L. D. F. Costa, J. Oliveira, Osvaldo, G. Travieso, F. A. Rodrigues, P. Villas Boas, L. Antiqueira, M. P. Viana, L. Correa Rocha, Analyzing and modeling real-world phenomena with complex networks: a survey of applications, Advances in Physics 60 (3) (2011) 329–412.

[9] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, M. A. Porter, Multilayer networks, Journal of complex networks 2 (3) (2014) 203–271.

[10] R. Gallotti, M. Barthelemy, The multilayer temporal network of public transport in great britain, Scientific data 2 (1) (2015) 1–8.

[11] A. Aleta, S. Meloni, Y. Moreno, A multilayer perspective for the analysis of urban transportation systems, Scientific reports 7 (1) (2017) 44359.

[12] C. T. Ba, A. Michienzi, B. Guidi, M. Zignani, L. Ricci, S. Gaito, Fork-based user migration in blockchain online social media, in: 14th ACM Web Science Conference 2022, 2022, pp. 174–184.

[13] B. Lee, S. Zhang, A. Poleksic, L. Xie, Heterogeneous multi-layered network model for omics data integration and analysis, Frontiers in genetics 10 (2020) 1381.

[14] P. Perlasca, M. Frasca, C. T. Ba, J. Gliozzo, M. Notaro, M. Pennacchioni, G. Valentini, M. Mesiti, Multi-resolution visualization and analysis of biomolecular networks through hierarchical community detection and web-based graphical tools, PloS one 15 (12) (2020) e0244241.

[15] A. Galdeman, C. Ba, M. Zignani, S. Gaito, A multilayer network perspective on customer segmentation through cashless payment data, in: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2021, pp. 1–10.

[16] R. Interdonato, M. Magnani, D. Perna, A. Tagarelli, D. Vega, Multilayer network simplification: approaches, models and methods, Comput. Sci. Rev. 36 (2020) 100246.

[17] E. O. Laumann, P. V. Marsden, D. Prensky, The boundary specification problem in network analysis, Research methods in social network analysis 61 (8) (1989).

[18] G. Kossinets, Effects of missing data in social networks, Social networks 28 (3) (2006) 247–268.

[19] R. Sharma, M. Magnani, D. Montesi, Investigating the types and effects of missing data in multilayer networks, in: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, 2015, pp. 392–399.

[20] M. E. Dickison, M. Magnani, L. Rossi, Multilayer social networks, Cambridge University Press, 2016.

[21] G. L. Robins, Doing social network research: Network-based research design for social scientists, Doing Social Network Research (2015) 1–280.

[22] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, W. Wang, Robust graph representation learning via neural sparsification, in: ICML, 2020.

[23] R. Wickman, X. Zhang, W. Li, Sparrl: Graph sparsification via deep reinforcement learning, ArXiv abs/2112.01565 (2021).

[24] U. S. Shanthamallu, J. J. Thiagarajan, H. Song, A. Spanias, Gramme: Semisupervised learning using multilayered graph attention models, IEEE Trans. Neural Networks Learn. Syst. 31 (10) (2020) 3977–3988. doi:10.1109/TNNLS.2019.2948797.
URL https://doi.org/10.1109/TNNLS.2019.2948797

[25] L. Zangari, R. Interdonato, A. Calió, A. Tagarelli, Graph convolutional and attention models for entity classification in multilayer networks, Applied Network Science 6 (1) (2021) 1–36.

[26] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).

[27] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in neural information processing systems 31 (2018).

[28] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: International conference on machine learning, PMLR, 2019, pp. 7134–7143.

[29] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, C. Wang, Hierarchical graph pooling with structure learning, arXiv preprint arXiv:1911.05954 (2019).

[30] S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui, Graph neural networks in recommender systems: A survey, ACM Comput. Surv. 55 (5) (dec 2022). doi:10.1145/3535101.
URL https://doi.org/10.1145/3535101

[31] M. Dileo, M. Zignani, S. Gaito, Temporal graph learning for dynamic link

prediction with text in online social networks, Machine Learning (Nov 2023). doi:10.1007/s10994-023-06475-x.
URL https://doi.org/10.1007/s10994-023-06475-x
[32] M. Zitnik, M. Agrawal, J. Leskovec, Modeling polypharmacy side effects with graph convolutional networks, Bioinformatics 34 (13) (2018) 457–466.
[33] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
[34] J. Li, C. Chen, H. Tong, H. Liu, Multi-layered network embedding, in: Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, 2018, pp. 684–692.
[35] M. Ghorbani, M. S. Baghshah, H. R. Rabiee, Mgcn: semi-supervised classification in multi-layer graphs with graph convolutional networks, in: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2019, pp. 208–211.
[36] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: Towards deep graph convolutional networks on node classification, in: ICLR, 2020.
[37] Y. Liu, T. Safavi, A. Dighe, D. Koutra, Graph summarization methods and applications: A survey, ACM computing surveys (CSUR) 51 (3) (2018) 1–34.
[38] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[39] W. L. Hamilton, Graph representation learning, Morgan & Claypool Publishers, 2020.
[40] E. Jang, S. S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, ArXiv abs/1611.01144 (2017).
[41] C. J. Maddison, A. Mnih, Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, ArXiv abs/1611.00712 (2017).
[42] E. Benjaminson, The Gumbel-Softmax Distribution.
URL https://sassafras13.github.io/GumbelSoftmax/
[43] D. Li, T. Yang, L. Du, Z. He, L. Jiang, Adaptivegcn: Efficient gcn through adaptively sparsifying graphs, Proceedings of the 30th ACM International Conference on Information & Knowledge Management (2021).
[44] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, X. Zhang, Learning to drop: Robust graph neural network via topological denoising, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 779–787.
[45] L. Wang, W. Yu, W. Wang, W. Cheng, W. Zhang, H. Zha, X. feng He, H. Chen, Learning robust representations with graph denoising policy network, 2019 IEEE International Conference on Data Mining (ICDM) (2019) 1378–1383.
[46] J. You, T. Du, J. Leskovec, Roland: graph learning framework for dynamic graphs, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2358–2366.
[47] C. T. Ba, M. Zignani, S. Gaito, The role of cryptocurrency in the dynamics of blockchain-based social networks: The case of steemit, PloS one 17 (6) (2022) e0267612.
[48] C. T. Ba, M. Zignani, S. Gaito, G. P. Rossi, The effect of cryptocurrency price on a blockchain-based social network, in: Complex Networks & Their Applications IX, Springer International Publishing, Cham, 2021, pp. 581–592.
[49] C. T. Ba, M. Zignani, S. Gaito, The role of groups in a user migration across blockchain-based online social media, in: 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), IEEE, 2022, pp. 291–296.
[50] C. T. Ba, A. Galdeman, M. Dileo, M. Zignani, S. Gaito, User migration prediction in blockchain socioeconomic networks using graph neural networks, in: Proceedings of the 2023 ACM Conference on Information Technology for Social Good, GoodIT '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 333–341. doi:10.1145/3582515.3609552.
URL https://doi.org/10.1145/3582515.3609552
[51] S. Kumar, R. Zafarani, H. Liu, Understanding user migration patterns in social media, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 25, 2011, pp. 1204–1209.
[52] E. Newell, D. Jurgens, H. Saleem, H. Vala, J. Sassine, C. Armstrong, D. Ruths, User migration in online social networks: A case study on reddit during a period of community unrest, in: Proceedings of the International AAAI Conference on Web and Social Media, Vol. 10, 2016, pp. 279–288.
[53] M. Senaweera, R. Dissanayake, N. Chamindi, A. Shyamalal, C. Elvitigala, S. Horawalavithana, P. Wijesekara, K. Gunawardana, M. Wickramasinghe, C. Keppitiyagama, A weighted network analysis of user migrations in a social network, in: 2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer), IEEE, 2018, pp. 357–362.
[54] C. Davies, J. Ashford, L. Espinosa-Anke, A. Preece, L. Turner, R. Whitaker, M. Srivatsa, D. Felmlee, Multi-scale user migration on reddit, AAAI, 2021.
[55] H. B. Zia, J. He, A. Raman, I. Castro, N. Sastry, G. Tyson, Flocking to mastodon: Tracking the great twitter migration, arXiv preprint arXiv:2302.14294 (2023).
[56] L. L. Cava, L. M. Aiello, A. Tagarelli, Get out of the nest! drivers of social influence in the #twittermigration to mastodon (2023). arXiv:2305.19056.
[57] A. Galdeman, M. Zignani, S. Gaito, User migration across web3 online social networks: behaviors and influence of hubs, in: In Proceedings of IEEE International Conference on Communications, 2023 (Accepted).
[58] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Tech. rep., Stanford infolab (1999).
[59] L. Martirano, L. Zangari, A. Tagarelli, Co-mlhan: contrastive learning for multilayer heterogeneous attributed networks, Applied Network Science 7 (1) (2022) 1–44.
[60] A. Breitfuss, K. Errou, A. Kurteva, A. Fensel, Representing emotions with knowledge graphs for movie recommendations, Future Generation Computer Systems 125 (2021) 715–725.
[61] T. Ma, H. Wang, L. Zhang, Y. Tian, N. Al-Nabhan, Graph classification based on structural features of significant nodes and spatial convolutional neural networks, Neurocomputing 423 (2021) 639–650. doi:https://doi.org/10.1016/j.neucom.2020.10.060.
URL https://www.sciencedirect.com/science/article/pii/S0925231220316374
[62] F. Bianchi, C. Gallicchio, A. Micheli, Pyramidal reservoir graph neural network, Neurocomputing 470 (2022) 389–404. doi:https://doi.org/10.1016/j.neucom.2021.04.131.
URL https://www.sciencedirect.com/science/article/pii/S0925231221011152
[63] L. Martirano, D. Ienco, R. Interdonato, A. Tagarelli, Dyhane: dynamic heterogeneous attributed network embedding through experience node replay, Applied Network Science 9 (1) (2024) 1–28.
[64] R. Interdonato, R. Gaetano, D. L. Seen, M. Roche, G. Scarpa, Extracting multilayer networks from sentinel-2 satellite image time series, Network Science 8 (S1) (2020) S26–S42.
[65] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

Appendix A. Appendix

Appendix A.1. Analysis of simplified graphs - datasets not included in the paper

Table A.7: Statistics for each graph layer before and after the simplification on um-socioeco dataset.

		Intra-layer edges	Label Assortativity	Transitivity	Indegree mean	Indegree max	Outdegree mean	Outdegree max
MARA (NS)	L0	579352.00	0.06	0.20	130.25	1875.00	130.25	2990.00
		4624.00	0.14	0.00	4.02	31.00	4.02	5.00
		(-99.20%)	(+146.19%)	(-100.00%)	(-96.92%)	(-98.35%)	(-96.92%)	(-99.83%)
	L1	476439.00	0.05	0.11	107.64	1759.00	107.64	4262.00
		4652.00	0.15	0.01	4.02	34.00	4.02	6.00
		(-99.02%)	(+181.36%)	(-95.09%)	(-96.26%)	(-98.07%)	(-96.26%)	(-99.86%)
	L2	74580.00	0.12	0.01	19.38	2543.00	19.38	3753.00
		4603.00	0.44	0.03	4.01	331.00	4.01	5.00
		(-93.83%)	(+277.44%)	(+287.20%)	(-79.30%)	(-86.98%)	(-79.30%)	(-99.87%)
	L3	14856.00	0.39	0.01	6.26	276.00	6.26	701.00
		4586.00	0.73	0.00	4.01	66.00	4.01	5.00
		(-69.13%)	(+85.09%)	(-100.00%)	(-36.02%)	(-76.09%)	(-36.02%)	(-99.29%)
MARA (DE)	L0	579352.00	0.06	0.20	130.25	1875.00	130.25	2990.00
		492450.00	0.06	0.17	111.16	1601.00	111.16	2543.00
		(-15.00%)	(+0.84%)	(-15.26%)	(-14.65%)	(-14.61%)	(-14.65%)	(-14.95%)
	L1	476439.00	0.05	0.11	107.64	1759.00	107.64	4262.00
		404974.00	0.05	0.09	91.95	1493.00	91.95	3623.00
		(-15.00%)	(+0.57%)	(-15.12%)	(-14.58%)	(-15.12%)	(-14.58%)	(-14.99%)
	L2	74580.00	0.12	0.01	19.38	2543.00	19.38	3753.00
		63393.00	0.12	0.01	16.92	2173.00	16.92	3201.00
		(-15.00%)	(-0.89%)	(-14.71%)	(-12.68%)	(-14.55%)	(-12.68%)	(-14.71%)
	L3	14856.00	0.39	0.01	6.26	276.00	6.26	701.00
		12628.00	0.39	0.01	5.77	229.00	5.77	604.00
		(-15.00%)	(+0.24%)	(-14.64%)	(-7.81%)	(-17.03%)	(-7.81%)	(-13.84%)

Table A.8: Statistics for each graph layer before and after the simplification on Koumbia 5 dataset.

		Intra-layer edges	Label Assortativity	Transitivity	Indegree mean	Indegree max	Outdegree mean	Outdegree max
MARA (NS)	L0	4157.00	0.84	0.25	7.15	33.00	7.15	38.00
		2252.00	0.95	0.00	6.30	28.00	6.30	28.00
		(-45.83%)	(+12.85%)	(-100.00%)	(-11.87%)	(-15.15%)	(-11.87%)	(-26.32%)
	L1	5752.00	0.70	0.22	9.03	39.00	9.03	36.00
		2249.00	0.90	0.00	7.47	27.00	7.47	25.00
		(-60.90%)	(+28.24%)	(-100.00%)	(-17.27%)	(-30.77%)	(-17.27%)	(-30.56%)
	L2	4951.00	0.70	0.22	8.64	47.00	8.64	53.00
		2252.00	0.88	0.00	7.44	41.00	7.44	41.00
		(-54.51%)	(+25.94%)	(-100.00%)	(-13.91%)	(-12.77%)	(-13.91%)	(-22.64%)
	L3	3635.00	0.98	0.24	6.82	39.00	6.82	42.00
		2252.00	1.00	0.00	6.21	37.00	6.21	36.00
		(-38.05%)	(+1.36%)	(-100.00%)	(-9.02%)	(-5.13%)	(-9.02%)	(-14.29%)
	L4	5605.00	0.68	0.20	9.29	48.00	9.29	50.00
		2266.00	0.87	0.04	7.80	41.00	7.80	41.00
		(-59.57%)	(+28.12%)	(-79.11%)	(-16.00%)	(-14.58%)	(-16.00%)	(-18.00%)
MARA (DE)	L0	4157.00	0.84	0.25	7.15	33.00	7.15	38.00
		3534.00	0.85	0.20	6.87	33.00	6.87	37.00
		(-14.99%)	(+0.65%)	(-18.32%)	(-3.88%)	(-)	(-3.88%)	(-2.63%)
	L1	5752.00	0.70	0.22	9.03	39.00	9.03	36.00
		4890.00	0.70	0.19	8.65	36.00	8.65	35.00
		(-14.99%)	(+0.28%)	(-16.06%)	(-4.25%)	(-7.69%)	(-4.25%)	(-2.78%)
	L2	4951.00	0.70	0.22	8.64	47.00	8.64	53.00
		4209.00	0.69	0.19	8.31	46.00	8.31	52.00
		(-14.99%)	(-1.12%)	(-13.45%)	(-3.82%)	(-2.13%)	(-3.82%)	(-1.89%)
	L3	3635.00	0.98	0.24	6.82	39.00	6.82	42.00
		3090.00	0.98	0.20	6.58	39.00	6.58	40.00
		(-14.99%)	(+0.18%)	(-14.39%)	(-3.56%)	(-)	(-3.56%)	(-4.76%)
	L4	5605.00	0.68	0.20	9.29	48.00	9.29	50.00
		4765.00	0.68	0.17	8.92	47.00	8.92	48.00
		(-14.99%)	(-0.38%)	(-14.34%)	(-4.03%)	(-2.08%)	(-4.03%)	(-4.00%)

```
{'datasets': [('um-econ', 'features'),
  ('um-socioeco', 'features'),
  ('imdb-mlh', 'features'),
  ('Koumbia_2', 'features'),
  ('Koumbia_5', 'features')],
 'architecture': ['multi'],
 'architecture_simp': ['multi', 'single'],
 'model': ['gcn', 'gcn-de', 'gcn-ns'],
 'gnn_level': [True, False],
 'drop_rate_p': [0.1, 0.3, 0.5, 0.7],
 'k': [5, 10, 15],
 'tau': [0.001],
 'standardize': [True],
 'feat-variability': ['fixed'],
 'split': ['25 50 25'],
 'plots': [True],
 'early-stop': [True],
 'fastmode': [True],
 'gpu': [1],
 'run': [1],
 'debugging': [False],
 'dropout': [0.3],
 'hidden': [16, 32],
 'lr': [0.002],
 'num-layers': [2],
 'ns_num_hidden': [32],
 'epochs': [250],
 'patience': [10]}
```

Cheick Tidiane Ba is a Research Assistant at Queen Mary University London, with research focused on network science. He received a Ph.D. in Computer Science at Università degli Studi di Milano Statale after working at the Connets Lab in the Department of Computer Science. His research activity takes place within data and network science, focusing on data mining and machine learning applications on different networked systems, from blockchain to online social networks. He has published works on network analysis and modeling, machine learning on networks, as well as networks for bioinformatics.

Dino Ienco received the M.Sc. and Ph.D. degrees in computer science both from the University of Torino, Torino, Italy, in 2006 and 2010, respectively. He joined the TETIS Laboratory, IRSTEA, Montpellier, France, in 2011 as a Junior Researcher. His main research interests include machine learning, data science, social media analysis, information retrieval and spatio-temporal data analysis with a particular emphasis on remote sensing data and Earth Observation data fusion. Dr. Ienco served on the program committee of many international conferences on data mining, machine learning, and databases including IEEE ICDM, ECML PKDD, ACML, and IJCAI as well as served as a Reviewer for many international journals in the general field of data science and remote sensing.

Roberto Interdonato received a Ph.D. degree in computer engineering from the University of Calabria, Arcavacata, Italy, in 2015. His Ph.D. dissertation was titled "novel ranking problems in information networks". He is currently a Research Scientist with CIRAD, UMR TETIS, Montpellier, France. He was previously a Postdoctoral Researcher with the University of La Rochelle, La Rochelle, France, Uppsala University, Uppsala, Sweden, and the University of Calabria, Arcavacata, Italy, His research interests include the design of data science techniques applied to the analysis of complex networks (e.g., social media networks, trust networks, semantic networks, bibliographic networks) and the extraction of information from remote sensing data. His most recent contributions concern the implementation of deep learning methods for land use classification based on the analysis of time series of multisensor satellite images (optical, radar, high/very high spatial resolution), the application of complex network analysis techniques for the extraction of spatialized indicators (landscape, socio-economic) from multisource data (remote sensing, survey data, statistics, social networks, etc.). His thematic interests mainly concern the characterization of tropical agricultural landscapes, the production of spatial information for food security and the analysis of the transnational land trade market. On these topics, he has co-authored journal articles and conference papers, organized workshops, presented tutorials at international conferences and developed practical software tools.

Sabrina Gaito is the director of the Connets Lab in the Department of Computer Science at the University of Milan, where she teaches social network analysis and machine learning. Her research activity takes place within data and network science, focusing on social networking and human mobility.