This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

On Inductive–Transductive Learning with Graph Neural Networks

Giorgio Ciano, Alberto Rossi, Monica Bianchini, and Franco Scarselli,

***Abstract*—Many real–world domains involve information naturally represented by graphs, where nodes denote basic patterns while edges stand for relationships among them. The Graph Neural Network (GNN) is a machine learning model capable of directly managing graph–structured data. In the original framework, GNNs are inductively trained, adapting their parameters based on a supervised learning environment. However, GNNs can also take advantage of transductive learning, thanks to the natural way they make information flow and spread across the graph, using relationships among patterns. In this paper, we propose a mixed inductive–transductive GNN model, study its properties and introduce an experimental strategy that allows us to understand and distinguish the role of inductive and transductive learning. The preliminary experimental results show interesting properties for the mixed model, highlighting how the peculiarities of the problems and the data can impact on the two learning strategies.**

***Index Terms*—Graph Neural Networks, Transductive Learning, Inductive learning.**

I. Introduction

GRAPHS are powerful and versatile data structures and constitute a natural way of representing information coming, for instance, from social networks, cybersecurity, and computational biology. The main advantage of graphs is that they easily allow to represent entities (nodes) and interactions among them (edges), possibly attaching further information on the nature of the existing relationships. For example, various types of interaction networks (metabolic, signaling and transcription–regulatory networks) — which form a network of networks — are responsible for the cell behavior [1]. Understanding how protein and gene regulatory networks differ between healthy and sick cells, and/or in response to known or novel drugs, has the potential to enhance drug development efforts to identify new targets, and to predict off–target effects, also enabling individual care [2]. Topological and dynamical characteristics of biological neural networks highlight the structural and functional properties of the brain [3], which can help both neuroscientists and AI researchers in understanding cognitive mechanisms. Generally speaking, graph data are widespread in most real–world applications, where existing relations between basic information entities cannot be ignored without affecting the very nature of the problem.

In the last two decades, several machine learning approaches, capable of processing graph–structured data have been proposed, based on support vector machines [4]–[9], random fields [10], statistical relational learning [11], transductive and semi–supervised learning [12], [13], and artificial neural networks. In particular, neural network models for graphs have been devised both in the supervised [14], [15] and in the unsupervised framework [16], [17] since the late '90s. Recursive Neural Networks (RNNs) [14], [15] are the forefather of supervised approaches. RNNs process directed positional acyclic graphs, taking into account both symbolic and subsymbolic information — collected into the node labels and defined by the graph topology, respectively — to calculate a set of states associated with the graph nodes. The states are dynamically updated following the graph topology, and the output at each node is computed based on a *graph encoding* stored in the state. Graph Neural Networks (GNNs) [18], instead, can process input data encoded as general labeled graphs. The state at each node is iteratively evaluated based on local information solely, realizing a sort of data diffusion mechanism across the whole graph. Moreover, GNNs are provided with a supervised learning algorithm that, besides the standard input–output data fitting cost, incorporates a criterion aimed at enforcing a contractive dynamics, to ensure the convergence of the state computation also for cyclic graphs. Both node–focused and graph–focused[1] problems can be addressed by GNNs, meaning that an output is produced for each node or for a unique node of the graph.

Recently, the interest in connectionist models for graphs has rapidly raised, and a large number of GNN variants have been proposed. For instance, in [19], the inclusion of gated recurrent units is proposed to extend GNNs to output sequences. The GraphSAGE model [20] exploits a function that generates embeddings by sampling and aggregating features from node local neighborhoods. In [21], a general model, which also covers GNNs, has been proposed based on the idea of message passing among nodes. Moreover, a huge research activity was devoted to generalize convolutional networks from regular grids to high–dimensional irregular domains, represented by graphs [22]–[26]. The outcome of this research was two–fold: a spectral formulation was introduced, allowing a deeper theoretical understanding of new approaches; novel models were proposed — in this context, the original GNN is just an example of such models. Graph Convolutional Networks (GCNs) [26] represent a seminal example of this research. GCNs process graphs using convolutional layers and find their theoretical foundation in the spectral graph theory. A review

G. Ciano and A. Rossi are with the University of Florence and with University of Siena, Italy, e-mail: giorgio.ciano@unifi.it and alberto.rossi@unifi.it, respectively.

M. Bianchini and F. Scarselli are with the University of Siena, Italy, e-mail: monica@diism.unisi.it and franco@diism.unisi.it, respectively.

G. Ciano and A. Rossi contributed equally to this work.

[1] A problem is said graph–focused if the GNN has to produce a single output for the entire graph, whereas, in node–focused problems, one output for each node is required.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

of neural network methods for graphs and, more generally, for non–Euclidean data, can be found in [27], [28].

Interestingly, GNNs and most of their descendants, can naturally exploit both inductive and transductive learning. In the inductive learning framework, a parametric model I_w is learnt by adjusting its weights w based on a training set. Then, the model can be applied to novel test patterns without further accessing the training set. With transductive learning (see [29]–[32]), instead, the training set patterns and their targets are used in conjunction with the test patterns. The decision on the test set is taken using a diffusion mechanism, e.g., exploiting the intuition that patterns with similar features are expected to be similar and belong to the same class.

The aim of this paper is to study the properties, together with advantages and limitations, of the two learning frameworks applied to GNNs. To this end, we propose a mixed inductive–transductive model that can reproduce the peculiarities of both the frameworks at the same time. This paradigm allows us to use the training patterns both as the source of the transduction, in transductive learning, and to train the network parameters, in inductive learning. The transductive learning paradigm in GNNs had already been proposed in [33] for the problem of spam detection on the Web. In this manuscript, however, we extensively evaluate its impact, in combination with inductive learning, to understand when and to what extent it is useful and able to add robustness to the GNN model.

To disentangle the contributions of the inductive and transductive parts of the model, we used an experimental methodology, based on the addition of noise on the node labels and on the repetition of experiments with different quantities of inductive–transductive patterns. The experiments were carried out using the original GNN model and the GCN model. The original GNN was chosen for its ability to approximate the set of functions on graphs that preserve the unfolding equivalence, and which represent the largest class of functions that can be implemented by a GNN based on local message passing [34][2]. This is a useful characteristic for the theoretical analysis that we want to carry out, since it ensures that the model is general and independent from design peculiarities. Instead, GCNs are chosen because they are simple and well–known architectures, whose performance is, however, usually not far from the state–of–the-art.

Several experiments have been conducted on synthetic and real datasets, involving structured inputs encoded as graphs. The results revealed interesting properties of the inductive–transductive model and have suggested that it could provide some advantages over the original inductive model. Furthermore, the experiments evidenced interesting examples of conditions when one of the two parts, the inductive or the transductive component, is predominant on the other. These conditions may depend on the data characteristics, e.g. groups of nodes which are "clustered" within the graph, or on the problem peculiarity, e.g. its complexity.

The rest of the paper is organized as follows. In Section II, the GNN and the GCN models, together with the related learning algorithms, are briefly sketched and the mixed inductive–transductive learning framework is described. Section III presents the experimental settings and reports the obtained results. Finally, Section IV draws some conclusions and delineates future perspectives.

II. Neural Network Methods for Graphs

A. The Graph Neural Network Model

A graph G is defined as a pair $G = (V, E)$, where V represents a finite set of *nodes* and $E \subseteq V \times V$ denotes a set of *edges*. We assume the edges are undirected, i.e. the pair of nodes constituting the edge is not ordered, $(a, b) = (b, a)$[3]. Both edges and nodes can be enriched by attributes that are collected into feature vectors. In the following, we consider feature vectors of a predefined size, eventually different for nodes and edges.

The Graph Neural Network is a supervised architecture able to face classification and regression tasks, where inputs are encoded as graphs [18]. The computation is driven by the input graph topology. To each node, a state vector is attached, which is updated as a function of the node label and of the informative contribution of its neighborhood (edge labels and states of the neighboring nodes, see Fig. 1), based on an *information diffusion mechanism*. Indeed, GNNs are supposed to capture the local information relevant to the given task — which is stored into the node state — and, finally, thanks to the diffusion process, to collect the whole information attached to the input graph. Afterward, the state is used to compute the node output, f.i. its class or a target property.

More formally, let $x_n(t) \in \mathbb{R}^s$ and $o_n(t) \in \mathbb{R}^m$ be the state and the output of node n at time t, respectively. Then, the computation locally performed at each node during the diffusion process can be described by the following equations

$$x_n(t+1) = \sum_{v:(n,v)\in E} f_w(l_n, l_{(n,v)}, x_v(t), l_v)\,, \qquad (1)$$

$$o_n(t+1) = g_w(x_n(t), l_n)\,, \qquad (2)$$

where f_w is the state transition function, which drives the diffusion process, while g_w represents the output function. Moreover, $l_n \in \mathbb{R}^q$, and $l_{(n,v)} \in \mathbb{R}^p$ are the labels attached to n and (n, v), respectively. As previously stated, the computation considers the neighborhood of n, defined by its edges $(n, v) \in E$. In particular, for each node v adjacent to n, the state x_v and the label l_v are used in the state and (indirectly) in the output calculation (Fig. 1). The summation in Eq. (1) allows us to deal with any number of neighbors without specifying a particular position for each of them.

Usually, both f_w and g_w are implemented by multilayer perceptrons, with a single hidden layer. Eq. (1) is replicated

[2]In [34], it is proved that the GNN model can approximate in probability up to any degree of precision any function that preserves the unfolding equivalence. However, recently, researchers have used the Weisfeiler–Lehman test in place of the unfolding equivalence to analyse the computational capability of GNN variants and in [35] several recent models, e.g., GCN and GraphSAGE, are proved to be unable to pass such a test, so that they are not universal approximators as the original model.

[3]It is worth mentioning that GNNs can deal also with directed graphs. Here, for the sake of simplicity, we have chosen to consider only non–oriented edges.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

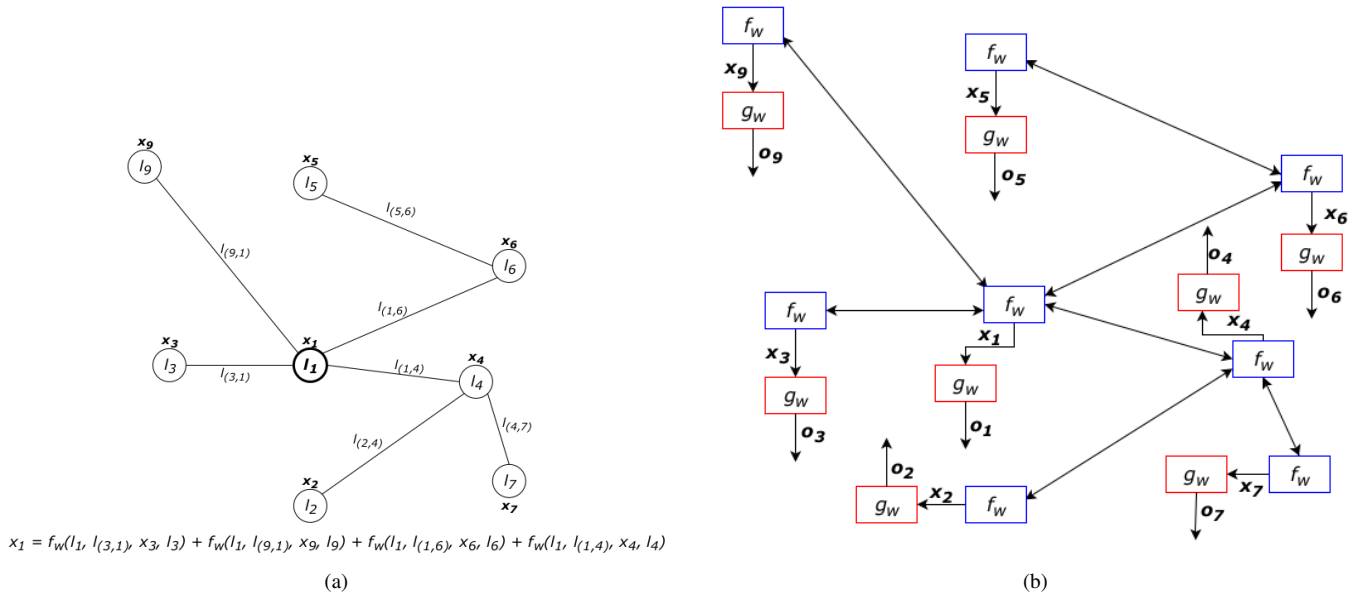


Fig. 1: Evaluation of the state transition function for node 1 (a). The corresponding encoding network (b).

on all the nodes of the graph and defines a non–linear dynamic system that describes the unfolding of the *encoding network* (Fig. 1). Actually, the encoding network is a recurrent network such that, for each node of the input graph, two modules exist: the f_w module, which is in charge of computing the node state; the g_w module, which calculates the node output. The connectivity in the encoding network, namely how the f_w modules exchange the node states, depends on the graph connectivity.

In order to guarantee the convergence to a steady state in Eqs. (1)–(2), the original GNN model forced the dynamics of the system to be contractive [18]. In this case, the Banach Theorem ensures the existence of a unique fixed point and the independence of such a fixed point from the initial state. During training, the network weights are adapted to reduce the error between the network outputs and the expected targets on a set of supervised nodes, namely nodes for which the desired output is available. The gradient computation is performed applying the BackPropagation Through Time algorithm on the unfolded encoding network, obtaining the so–called BackPropagation Through Structure (see [18], [36] for more details). More recent implementations of GNNs, f.i. [19], [37], relax the state convergence constraint, just computing the output after iterating Eq. (1) for a fixed number of steps. This demands more memory requirements for the gradient calculation with respect to [18], though it removes the need to constrain parameters to ensure convergence.

Under weak assumptions, the GNN model can approximate in probability all the functions on graphs with any required precision [34], showing a generalization capability similar to that of recurrent neural networks [38]. A recent theoretical study on GNN properties can be found in [35].

B. The Graph Convolutional Network

The Graph Convolutional Network (GCN) is a seminal model capable to process graphs. It is based on the idea of convolution, namely on the fact that activations associated to the nodes can be computed layer–wise by propagating the node labels through the graph. The GCN propagation rule is

$$X^{(k+1)} = \sigma(\hat{A}X^{(k)}W^{(k)}), \tag{3}$$

where the matrix $\hat{A}$ is a normalized version of the adjacency matrix of the graph (see [26], for more details), $X^{(k)}$ is the matrix of the activations, and $W^{(k)}$ is the set of of the k–th layer weights. Moreover, σ is the activation function, while the activation values at time 0 are all equal to the labels, i.e., $X^{(0)} = L$, where L is the matrix of the node labels. In our experiments, we will employ a two–layer version of this architecture, which has been previously used in [26]. In this case, the function implemented by the GCN with respect to the graph adjacency matrix $\hat{A}$ and the graph labels L is

$$\phi(\hat{A}, L) = \text{softmax}\left(\hat{\text{A}} \cdot \text{ReLu}\left(\hat{\text{A}}\text{LW}^{(0)}\right)\text{W}^{(1)}\right). \tag{4}$$

The main characteristics of inductive and transductive learning are examined below, before introducing the proposed mixed inductive–transductive approach.

C. Inductive vs transductive learning

In the common *inductive* learning framework, a model I_w is learned by adjusting its weights, w, based on a set of supervised examples, collected in the training set. The overall learning procedure is aimed at minimizing a suitable loss function that induces the model to capture the statistical distribution of training data. After training, the model I_w can be applied to new patterns, never seen before, completely neglecting the training set, whose related information is collected into the learned parameters w.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

Conversely, in the *transductive* framework, learning may not be based on any form of parameter tuning but, instead, both training and test examples can be exploited at the same time, taking advantage of their mutual relationships, such as, for instance, some spatial regularization in the feature space (e.g. manifold regularization). Relationships among data can be exploited either in the learning or in the prediction phase, or in both of them[4]. The prediction on the unsupervised data is obtained by propagating the information available on the neighboring examples, through a "diffusion mechanism" induced by the existing relations among patterns. For instance, if n is a test example, then the targets available on its neighboring patterns may be exploited as inputs — together with the local features of n — to compute its output.

This approach is particularly useful and natural when only a small set of supervised data, which comes from an unknown stochastic process, is available. Indeed, a small sample set cannot be statistically relevant for inducing a general predictive rule [39] based only on local features, which accounts for difficulties in applying a pure inductive framework. Consequently, in recent years, transductive learning techniques have been widely applied and implemented in many domains, thanks to their ability of being adaptable to different real–world applications, such as natural language processing [40], surveillance [41], graph reconstruction [42] and ECG classification [43].

It is worth noting that, in transductive learning, the information useful for processing a particular example is collected by exploring the examples related to it. For this reason, for plain data, the use of pattern relationships is often considered a distinctive feature of transductive learning. Nevertheless, the same characteristic cannot be used when the input domain is constituted by graphs. For relational data, indeed, the difference between the two frameworks must be defined focusing on how the training set targets are used. Thus, we can adopt the following definition: in inductive learning, the targets are used for tuning the parameters, whereas in transductive learning, they are used for the information diffusion.

D. The inductive–transductive framework

Modern neural network approaches to graph processing, including GNNs and the derived methods, are naturally prone to be used either for transductive and inductive learning. There even exist several examples in which researchers exploit the same model for both the two learning frameworks e.g., [20], [44], [45].

In order to understand, by simple experiments, advantages, disadvantages and peculiarities of the two learning approaches, and to clarify what happens when they are used in conjuction, we introduce a mixed inductive–transductive approach. In the proposed model, first the dataset is divided into training, validation and test sets. Then, for each of these sets, three disjoint sets of nodes are employed:

- The set of inductive nodes L, whose targets are used to compute the loss function and to adapt the parameters during the inductive network training;
- The set of transductive nodes T, whose targets are used in the transductive learning phase;
- The set of unsupervised nodes U, whose targets are not available.

The union of the sets of inductive and transductive nodes, $S = L \cup T$, constitutes the set of supervised nodes, for which a target is available.

In a pure inductive approach, only the inductive and the unsupervised sets of nodes exist. When graphs to be learnt are fed into the model, the targets of nodes in L are used only to learn the network parameters. The trained model can then be exploited to process both the original graph(s) in the learning set, or to compute the output for the unsupervised nodes in U. In other words, during learning, the model exploits only the graph topology and the information disseminated through the graph. Once the model has been trained, it can be used to generalize to unsupervised patterns; even in this phase, the prediction is based only on the node labels and on the graph topology, without any knowledge of the neighboring node targets.

For example, consider the Web Spam detection problem, in which the input is the Web graph, and the goal consists in classifying Web pages (the nodes of the graph) as spam or non–spam. In this case, L contains the pages whose target is known, used to adapt the network parameters. During the test phase, the targets of L are not used, since it is assumed that the trained network already embeds the classification rules needed to solve the problem[5].

Instead, in the mixed inductive–transductive learning framework, both the inductive and the transductive set of nodes are taken into account for learning. In particular, the labels of the nodes in T are enriched with their targets, to be explicitly exploited in the diffusion process, yielding a direct transductive contribution. Conversely, for the nodes in L and U, a special `null` target is attached (f.i. a vector of zeros). Formally, for a node n, the enriched label $\bar{l}_n$ will be

$$\bar{l}_n = \begin{cases} [l_n, t_n, 1] & if \;\; n \in T \\ [l_n, 0, 0] & if \;\; n \in U \cup L, \end{cases} \tag{5}$$

where the last scalar value of the label defines whether the node is transductive or not. Moreover, the targets of the inductive nodes in L are used to define the training loss, for tuning the network parameters.

Thus, in the mixed framework, the model has to learn to diffuse the information provided by the targets of the transductive nodes[6], which must be combined with the information coming from the node labels and from the graph topology.

We can observe that, given a set of supervised nodes S, different training environments for the inductive–transductive

[4]In fact, in most of the transductive algorithms, there are not two distinct phases for training and testing, but there is a single learning phase that serves both to adjust the possible parameters and to predict the targets on the test patterns.

[5]Note that, even if we cannot use the targets in L, we can still use the nodes of L during the test phase. For example, we can use the same Web graph both for training and testing the network.

[6]It is worth noting that, during training, L and T must remain distinct — otherwise the network would easily learn to produce the correct output at a particular node by taking the value t_n from the node label $\bar{l}_n$.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

model can be generated by different splits of S into L and T. The splits can be randomly generated, as long as the obtained L and U are representative of the actual distribution of supervised/unsupervised nodes in the graphs used for testing.

During the test phase, a set of transductive and unsupervised nodes are required for a correct working of the inductive–transductive model (with node labels defined as in Eq. (5)). The inductive nodes are not required for testing[7]. In practical applications, all the transduction targets could be used in order to maximize the available information. Thus, for example, it may be reasonable to include in T all the supervised nodes of the training set, provided that they are related to the test nodes[8].

Referring to the above mentioned Web Spam example, suppose that a training set of pages is available for which the targets are known. In order to apply the mixed inductive–transductive GNN approach, pages are randomly split into two sets, L and T, with labels extended as in Eq. (5). After training, the network can be tested on the entire Web graph, with T that includes also all the pages originally part of the training set.

It is worth mentioning that, in this approach, transduction is implemented by adding the target to the node features. This mechanism differs from those used in other approaches in which, usually, targets are directly employed in the optimization procedure.

III. Experimental Evaluation

In this section, an experimental study on the inductive–transductive learning approach for GNNs and GCNs is carried out. In particular, we first describe the employed datasets and the experimental setup. Then, we describe the strategy implemented to evaluate the contribution of the information diffusion process, showing and discussing the obtained results.

A. Datasets

The experiments have been carried out on two synthetic benchmarks, here called *subgraph matching* and *distance from the source*, and on four real–world datasets, namely *Web Spam*, *Cora*, *Citeseer* and *ogb–Arxiv*. In both synthetic benchmarks, data consist of randomly generated graphs. The goal of the subgraph matching problem is that of localizing a given subgraph inside a larger graph, while the objective of the distance from the source problem is that of computing the minimum distance between each node and a given target node, in terms of the number of arcs to traverse to go from one to the other. Instead, the Web Spam benchmark consists of a subset of the Web graph on which Web hosts have to be classified as spam or non–spam. Cora and Citeseer are datasets related to citation networks for scientific publications, widely used for testing graph based algorithms. Finally, the ogb–Arxiv dataset represents the citation network of all Computer Science (CS) Arxiv papers indexed by Microsoft Academic Graph (MAG) [46]. More details for each problem are given in the following.

1) Subgraph matching: The subgraph matching problem consists in finding the nodes of a given subgraph $S = (V_s, E_s)$ in a larger graph $G = (V, E)$. In the experiments, the dataset contains randomly generated, undirected graphs. A smaller subgraph S, which was randomly generated in each trial, was inserted into each graph of the dataset. Thus, all the graphs in the dataset contain at least a copy of S, even if more copies might have been included by the random construction procedure[9]. All the nodes had integer labels in the range $[0, 10]$, which can help in localizing the subgraph, while the target $t(G, n)$ for a node n in a graph G is defined as

$$t(G,n) = \begin{cases} 1 & if \quad n \in V_s \\ -1 & if \quad n \notin V_s \end{cases} \tag{6}$$

In the base experiment, each graph G was made up by 30 nodes, and the subgraph S by 7 nodes. The dataset consists of a total of 900 graphs, accounting for 300 graphs for each of the training, validation and test sets. Training, validation and test sets were disjoint, even if the percentage of transductive nodes was the same for all the sets. The graphs were produced by generating each edge independently, with a predefined probability $\delta = 0.2$. Later, further experiments will be described, carried out by varying these parameters.

Finding a given subgraph within a graph is an interesting problem, encountered in several practical applications, such as object localization in images and detection of active molecules in chemical compounds. Moreover, it allows testing the capability of the GNN model to integrate the data coming from the node labels with the information related to the graph connectivity.

2) Distance from the source: In this benchmark, the problem is to compute the minimum distance of each node from a set of designed source nodes. Actually, the source nodes are not known in advance by the model, but they can be recognized by their node labels and learned by examples.

To construct the dataset, a set of random graphs has been generated, with node labels constituted by random n–dimensional integer vectors in the range $[0, 3]$. Then, two nodes are arbitrarily chosen in each graph, to be signed as sources by replacing their original label with a particular label l_s: such a label, which is randomly generated only once for each dataset, is supposed to help the GNN in localizing the sources. The target $t(G, n)$ was formally defined as the number of edges in the shortest path from node n to any node with the label l_s.

In this benchmark, the training, validation and test sets contain 100 graphs, with each graph formed by 100 nodes. Also in this case, training, validation and test sets were disjoint, while the percentage of transductive nodes was the same for all the sets. The graphs were generated using $\delta = 0.9$ as the edge probability. Such a value was experimentally chosen to produce a good balance between two fundamental

[7]Obviously, some test nodes may still be used in place of the inductive nodes for evaluating the performance of the network.

[8]Even if the usage of the entire set S during the test phase can help the prediction, the implementation of such a procedure is difficult and will be a matter of future research. By deciding, at the test time, that T contains all the supervised nodes S, we implicitly define what is the relative proportion of the T nodes with respect to the other nodes. Therefore, such a proportion must be recreated in the training set, in order to make the GNN able to learn and generalize. Changing the proportion after the GNN is trained can create a bias for the model, compromising its prediction ability.

[9]The possible further copies have been localized by a brute force algorithm.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

requirements: the generated graphs must be almost fully connected, so that the source nodes can be reached easily from the other nodes; the paths that lead to the source nodes must be long and varied, otherwise the problem is too easy to be solved[10]. Using the above described parameters (δ and the number of nodes per graph), we found that targets usually take values in [0,15].

The distance from the source problem is interesting because of the behavior of the GNN, which produces a sort of recursive solution, where the output computed at a particular node is used to compute the output on its neighbor nodes. In the experiments, we will show how such behavior allows exploiting the advantages of the transductive learning framework.

3) Web Spam: The WEBSPAM–UK 2006 dataset [48] was obtained using a large set of .UK pages downloaded in May 2006 by the Laboratory of Web Algorithmics of the University of Milan. The dataset collects in a unique, and quite well–clustered graph, 11402 hosts and 730775 edges (links). Different types of features are present in the dataset, divided into three main categories — basic, link–based, and content–based. For our experiments, we used the ten most informative content–based features[11], which have been selected using decision trees.

The aim of the dataset is to classify hosts as spam or non–spam[12]. In the graph, each node represents a host and the node label consists of an array containing the above mentioned features. Differently from other benchmarks, directed arcs are used instead of undirected edges. More precisely, for each hyperlink of the Web which connects a host $h1$ to a host $h2$, there are two labeled arcs, $(h1, h2)$ and $(h2, h1)$. The two arcs have a different label, 1 and 0, respectively, to make the GNN aware on whether the arc has the direction of the hyperlink or not.

For the experiments, the original 6552 hosts have been split with a percentage of 70% for training, 10% for validation, and 20% for testing purposes.

The Web Spam dataset provides a useful testbed to evaluate the inductive–transductive GNN model, since it corresponds to a simple real–life application in which both the types of learning have been used in the past. Moreover, a well–known peculiarity of the Web Spam dataset is that spam pages tend to refer each other, whereas it is rare that a non–spam page has hyperlinks to spam pages [49]. The experiments actually show that such an asymmetry may play an interesting role in transductive learning.

4) Cora & Citeseer: The Cora and Citeseer datasets represent networks of papers in the field of machine learning categorized in seven and six classes, respectively. The node labels denote, by a one–hot representation, the words contained in the documents. The links represent the citations between the papers. In particular, Cora consists of 2708 documents, and has a vocabulary size of 1433 words and 5429 links, while Citeseer is made by 3312 documents, and has a vocabulary size of 3703 and 4732 links. In order to be able to run a large number of experiments, we decided to reduce the dimension of the node labels by a feature selection algorithm, based on decision trees, which produced 10 features[13]. Moreover, in Cora, 1000, 140, 500 nodes have been used for training, validation and testing, respectively, while in Citeseer, 1000, 120, 500 nodes have analogously been employed. Note that this splitting is not the original one, where the smallest set of nodes is used for training and the largest is exploited for testing. However, in our experimentation, we need more flexibility, since the number of transductive nodes are going to be varied. Moreover, we need to implement also the inductive part of the algorithm, which, differently from the pure transductive methods usually applied to Citeeseer and Cora, requires more patterns.

5) ogb–arXiv: The *ogb–arXiv* dataset [50] consists of a directed graph with 169,343 nodes, representing arXiv documents, and 1,166,243 edges, each of which indicates a citation. Each node is labelled with a 128–dimensional feature vector, obtained by averaging the embeddings of words in the title and abstract of the represented paper. There are 40 classes in this dataset that correspond to the subject areas of arXiv CS documents[14]. The task is to predict the main categories of the papers. Based on the year of publication of the articles, a division into training, validation, and test set was carried out. The papers published until 2017 belong to the training set, those published in 2018 represent the validation set and, finally, articles published from 2019 form the test set, obtaining 90,941, 29,799, 48,603 nodes for the training, validation and test set, respectively. For our experiments, both the original dataset splitting and the number of features are maintained.

B. Experimental Setup

All the experiments have been carried out using a GNN simulator written in Python[15] [37], except for the distance from the source benchmark, for which the original GNN simulator for MATLAB[16] has been employed. For the GCN, we used the model available online[17] from Kipf et.al. [26].

The network parameters (i.e., the state dimension and the number of hidden units for the GNN, the number of hidden units for the GCN, the number of epochs and the activation functions for both the architectures) have been chosen with a preliminary experimentation, consisting in the manual search

[10]The relationship between δ and the structure of the generated graphs is clarified in the theory of random graphs [47]. With small values of δ the generated graphs tend to be disconnected, whereas for large values of δ they are densely connected and have a very small diameter. Instead, when δ is close to $1/N$, the diameter of the generated structures tend to be maximal.

[11]Content–based features include statistical values measuring the distribution of words in the host pages as, for instance, the precision and the recall of words in a page with respect to the q most frequent terms from a query log, where $q = 100, 200, 500, 1000$.

[12]WEBSPAM–UK 2006 was originally tagged by a group of volunteer in three classes: “spam,” “non–spam” and “borderline.” In this paper, according to the common practice, non–spam hosts include both the borderline and the original non–spam class.

[13]Actually, the original GNN model requires a large amount of computational resources due to iterative convergence to a steady state and to the implementation of a transition function by a two layer neural network.

[14]https://arxiv.org/corr/subjectclasses

[15]The software is available online at http://sailab.diism.unisi.it/the-graph-neural-network-framework/

[16]The software is available online at http://www.dii.unisi.it/~franco/Research/GNNDownload.php/

[17]https://github.com/tkipf/gcn

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

of a set of parameters calibrated on the validation set without, however, trying to systematically maximize the performance of the models. In particular, for GNNs, the search started from our past experiences, whereas for GCNs, we started from the initial configuration of the available software[18]. Moreover, in all the experiments, neurons with the hyperbolic tangent activation function constitute the hidden layers of the GNN, while the softmax activation function is used for output neurons, except for the distance from the source problem, in which output linear neurons are employed.

Finally, the GNN architectures, implemented for the different benchmarks, the optimization techniques used for their training, and the number of epochs are detailed in the following.

- The network used for the subgraph matching problem has a state of dimension 5 and consists of a two layer transition network (with 15 and 5 neurons, respectively) and a two layer output network (with 10 hidden neurons and 2 output neurons). The GNN was trained for 1500 epochs using the Adam optimizer.
- The network used for the distance from the source problem has a state of dimension 2 and consists of a single hidden layer transition network (with 15 neurons) and a two layer output network (with 10 hidden neurons and 1 output neuron). The GNN was trained for 1000 epochs using the Resilient Propagation optimizer.
- For the Web Spam problem, the network has a state of dimension 5 and consists of a two layer transition network (with 10 and 5 neurons, respectively) and a single output layer with 2 neurons. The GNN was trained for 2000 epochs using the Adam optimizer.
- For the Cora dataset, the network has a state of dimension 10 and consists of a two layer transition network (with 10 and 5 neurons, respectively) and a two layer output network (with 10 hidden neurons and 7 output neurons). The GNN was trained for 4000 epochs using the Adam optimizer.
- For the Citeseer dataset, we used the same GNN as for Cora, except for the number of output neurons which is 6 and for the number of epochs wich is equal to 5000.

In all the cases, a full batch learning is used, meaning that we adapt the parameters once per epoch. This is straightforward in the case in which we have a unique graph to be learnt, whereas, when the training is related to a set of graphs, they are considered all together to form a single structure composed by disconnected components.

Regarding the GCN architecture, the number of hidden units is set to 16 for all the cases except for the Distance from the source and the Subgraph matching datasets, where 8 and 32 hiddens are used, respectively. The activation function is ReLu for all the datasets except for the Subgraph matching, in which the hyperbolic tangent is used. Morever, in all the cases, a dropout layer is used before the output layer. In the case of the Subgraph matching, a bias is enabled. The number of epochs was set to 2000 for both the Distance from the source and the Subgraph matching datasets, to 500 for Citeseer and Cora, and to 200 for WebSpam detection.

The results are always calculated by averaging the performance over five runs. The average accuracy, along with the corresponding standard deviation, are reported for all the datasets apart from the Distance fom the source benchmark, in which we measure the mean square error. Notice that the standard deviation mainly measures the sensitivity with respect to the choice of L and T, created randomly at each run, while the splitting of the dataset in training/validation/test sets was carried out once in each experiment[19].

Regarding the ogb–arXiv dataset, we will report the results achieved with two models from those mentioned in [50], namely GCN and GraphSAGE [20], using the same hyperparameters as in [50], i.e., 10 runs, 500 epochs, 256 hidden channels, 0.5 for the dropout, and 0.01 for the learning rate.

C. Experimental strategy

Intuitively, in order to study the properties of the proposed model, it is essential to distinguish between the inductive and the transductive contributions to its output. Notice that, actually, the mixed inductive–transductive paradigm aggregates two types of information, provided by the labels of the unsupervised nodes in U and by the targets of the transductive nodes in T, respectively. In the experiments, these two contributions are disentangled by two methods. On the one hand, several trials have been carried out using different percentages of unsupervised and transductive nodes. In this way, we can observe how the performance of the network changes when learning is transformed from pure inductive to increasingly transductive. On the other hand, experiments have been also carried out with an increasing quantity of noise, added to the labels of the unsupervised nodes. In fact, intuitively, by adding noise to node labels, we corrupt the information available to the inductive algorithm. In the extreme case, when the noise is very large, the information in the labels is lost and the inductive part has no information to use (except for the graph connectivity). Therefore, in this way, we can switch on and off the contribution coming from the inductive part of the GNN.

It is worth noting that the injection of noisy labels in synthetic datasets may be useful also to simulate real–life problems, where the information provided is rarely perfect, due to missing, incomplete or incorrect data and to the actual presence of noise.

D. Results and discussion

In the following, the experimental results are presented.

1) Subgraph matching: Tables I-II and Fig 2(a) show the results obtained on the subgraph matching problem using different percentages of transductive and unsupervised nodes and different noise levels. Four percentages of transductive

[18]Note that a more systematic hyperparameter selection procedure would have been difficult, since each experiment requires the optimization of the hyperparameters. Moreover, the goal was to find basic configurations able to evidence the behavior of the mixed approach, whereas a more comprehensive analysis of such parameters, tailored to maximize the model performance, is out of the scope of this work.

[19]More precisely, the standard deviation of each experiment includes also a variance due to the initialization of the learning algorithm, which, however, according to our experiments, tends to be a minor factor.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

Task	Noise	Transductive Percentage				
		0%	*10%*	*25%*	*50%*	*Diff.* 50% - 0%
SubGraph	*0%*	94.79 (0.88)	93.33 (0.5)	94.74 (0.87)	**95.56** (0.71)	0.77
	10%	84.63 (1)	85.01 (0.33)	86.11 (0.65)	**90.61** (0.83)	5.98
	25%	77.79 (0.17)	79.66 (0.41)	82.69 (0.31)	**87.43** (0.05)	9.64
	50%	76.48 (0.05)	77.31 (0.24)	81.28 (0.1)	**86.92** (0.21)	10.44
	100%	76.43 (0)	76.79 (0.44)	81.11 (0.46)	**86.71** (0.63)	10.28
WebSpam	*0%*	88.57 (0.99)	88.4 (1.37)	90.32 (5.53)	**92.36** (4.09)	3.79
	10%	86.58 (1.01)	88.41 (1.42)	89.02 (4.33)	**90.58** (5.29)	4
	25%	86.05 (0.71)	88.11 (1.06)	92.02 (1.06)	**94.87** (2.55)	8.82
	50%	84.9 (0.61)	86.5 (1.84)	88.68 (3.96)	**96** (1.39)	11.1
	100%	83.86 (0.76)	82.97 (1.64)	88.21 (2.81)	**92.91** (4.13)	9.05
Cora	*0%*	77.76 (1.71)	77.60 (1.65)	76.96 (2.48)	**79.20** (3.63)	1.44
	10%	76.48 (1.68)	75.16 (3.17)	73.76 (3.27)	**77.76** (3.94)	1.28
	25%	73.8 (2.84)	71.95 (5)	71.2 (3.87)	**76.32** (2.22)	2.52
	50%	66.64 (5.8)	68.04 (2.33)	65.44 (6.03)	**68.64** (4.39)	2
	100%	59.4 (4.31)	62.76 (6.53)	**64.05** (3.57)	62.96 (7.32)	3.56
Citeseer	*0%*	**69** (0.96)	68.40 (2.16)	67.73 (2.48)	65.68 (5.47)	-3.32
	10%	67.68 (1.21)	**68.13** (0.62)	66.56 (1.71)	66.08 (3.59)	-1.6
	25%	63.16 (3.12)	63.07 (2.28)	62.94 (2.17)	**65.04** (2.77)	1.88
	50%	53.4 (3.99)	55.42 (1.89)	56.85 (2.57)	**60.4** (3.14)	7
	100%	44.6 (2.06)	49.82 (4.25)	48.59 (7.85)	**54.32** (3.85)	9.72
Distance	*0%*	1.57 (0.01)	1.45 (0.08)	1.14 (0.05)	**0.94** (0.08)	-0.63
	10%	1.57 (0.01)	1.49 (0.05)	1.20 (0.08)	**0.95** (0.06)	-0.62
	25%	1.58 (0.01)	1.46 (0.07)	1.3 (0.11)	**0.95** (0.05)	-0.63
	50%	1.6 (0.01)	1.46 (0.06)	1.28 (0.08)	**1** (0.21)	-0.6
	100%	1.6 (0.01)	1.49 (0.03)	1.26 (0.04)	**1.05** (0.03)	-0.55

TABLE I: Performance of the inductive–transductive GNN. For the distance from the source problem, the MSE is used, while the accuracy results are described for the other datasets (the standard deviation is reported in brackets).

Task	Noise	Transductive Percentage				
		0%	*10%*	*25%*	*50%*	*Diff.* 50% - 0%
SubGraph	*0%*	79.44 (0.05)	**85.52** 0.84	81.88 (0.88)	79.76 (2.22)	0.32
	10%	77.08 (0.15)	**78.58** 0.62	76.98 (0.35)	74.62	-2.46
	25%	**76.14** (0.17)	75.32 (0.22)	75.22 (0.2)	73.58 (1.27)	-2.56
	50%	**75.98** (0.18)	75.48 (0.25)	74.7 (0.76)	74.7 (0.54)	-1.28
	100%	**75.86** (0.22)	75.56 (0.29)	74.8 (0.5)	73.96 (1.56)	-1.9
WebSpam	*0%*	80.2 (0)	82.6 (0)	82.4 (0)	**86.2** (0)	6
	10%	73.2 (0)	78.4 (0)	73.9 (0)	**84.2** (0)	11
	25%	73.2 (0)	76.2 (0)	**85.4** (0)	85.2 (0)	12
	50%	73.2 (0)	72.6 (0)	78.4 (0)	**86** (0)	12.8
	100%	73.2 (0)	74.5 (0)	73 (0)	**88.3** (0)	15.1
Cora	*0%*	73.6 (0)	72.9 (0)	76 (0)	**77.6** (0)	4
	10%	52 (0)	41.06 (0.09)	56 (0)	**67.12** (0.18)	15.12
	25%	**40.36** (0.09)	36.9 (0.14)	33.26 (0.09)	38.4 (0)	-1.96
	50%	32.52 (0.11)	32.9 (0)	26.66 (0.43)	**43.2** (0.89)	10.68
	100%	33.4 (0)	34.2 (0)	**35.2** (0)	32.96 (0.54)	-0.44
Citeseer	*0%*	62.32 (0.11)	62.56 (0.23)	66.08 (0.2)	**67.2** (0)	4.88
	10%	43.44 (0.54)	**45.84** (0.22)	44.9 (0.31)	43.36 (0.36)	-0.08
	25%	26.16 (0.09)	30.9 (0)	**40** (0.64)	30.4 (1.39)	4.24
	50%	19 (0)	**27.88** (0.18)	24 (0)	26.16 (0.22)	7.16
	100%	25.6 (0)	23.36 (0.13)	24.8 (0)	**26.4** (0)	0.8
Distance	*0%*	2.58 (0.01)	2.26 (0.08)	1.94 (0.21)	**1.09** (0.16)	-1.49
	10%	2.56 (0.05)	2.62 (0.67)	2 (0.38)	**1.6** (0.44)	-0.96
	25%	2.73 (0.26)	2.61 (0.29)	2.54 (0.22)	**1.78** (0.27)	-0.96
	50%	2.69 (0.03)	2.67 (0.16)	2.79 (0.72)	**2.46** (0.51)	-0.22
	100%	2.83 (0.24)	3.01 (2.84)	2.92 (0.27)	**2.81** (0.02)	-0.03

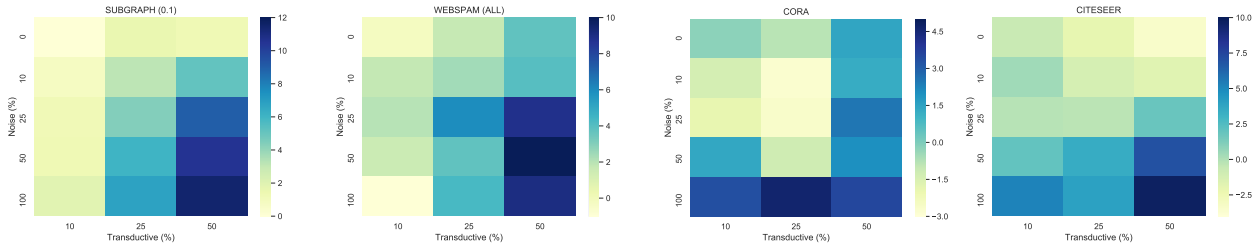
TABLE II: Performance of the inductive–transductive GCN. For the distance from the source problem, the MSE is used, while the accuracy results are described for the other datasets (the standard deviation is reported in brackets).

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

Density	Noise	Transductive Percentage				
		0%	*10%*	*25%*	*50%*	*Diff.* 50% - 0%
0.1	*0%*	93.87 (1.6)	93.98 (1)	**95.5** (0.64)	95.23 (1.31)	1.36
	10%	86.21 (0.60)	87.07 (1.05)	89.43 (0.80)	**91.38** (0.52)	5.17
	25%	81.35 (0.3)	82.65 (0.56)	85.8 (0.3)	**90.33** (0.3)	8.98
	50%	78.6 (0.3)	79.99 (0.29)	84.66 (0.74)	**89.11** (1.2)	10.51
	100%	77.01 (0.06)	79.01 (0.8)	83.82 (0.55)	**88.46** (1.15)	11.45
0.5	*0%*	91.34 (0.64)	90.04 (2.6)	**92.02** (1.18)	91.55 (0.95)	0.21
	10%	79.01 (0.44)	79.26 (0.28)	80.86 (0.99)	**85.8** (1)	6.79
	25%	76.56 (0.08)	77.84 (0.41)	80.28 (1.08)	**85.09** (0.49)	8.53
	50%	76.53 (0)	78.04 (0.16)	80.2 (1.75)	**83.2** (3.18)	6.67
	100%	76.53 (0)	76.51 (0.11)	80.56 (1.14)	**85.48** (0.08)	8.95

TABLE III: Accuracy scores of the GNN on the Subgraph dataset for each test (the standard deviation is reported in brackets) with different graph densities.



(a) (b) (c) (d)

Fig. 2: Difference between the performance of the inductive–transductive GNN and the pure inductive model.

nodes have been evaluated, namely 0% (corresponding to the pure inductive model), 10%, 25%, and 50%. Concerning the noise, we added a normal noise $N(\nu, \sigma)$ to each label, whose mean ν was set to 0 and whose variance σ ranged from 0 to a value proportional to the maximum label norm. In particular, we tested five configurations in which σ was set to 0%, 10%, 25%, 50%, and 100% of the maximum norm.

The experiments show how the transductive information allows to improve the GNN performance at each noise level, since the best accuracy is always achieved when the number of transductive nodes is maximal. Conversely, adding noise produces a general decrease in performance, for each percentage of transductive nodes, which suggests that a contribution from the inductive information also exists, rapidly deteriorated by noise. In summary, the obtained results suggest that the GNN is able to take into account and combine both types of information.

On the other hand, the GCN does not take advantage from the transductive nodes and, in general, the performance of the GCN is lower than that of the GNN in all the cases. Perhaps, this is due to the fact that the GCN model suffers from some limitations in terms of the graphs that it can distinguish [35] — a problem that does not affect GNNs. In the case of Subgraph matching, such a limitation is particularly important and cannot be alleviated by the transductive information.

It is worth noting that the importance of transductive information may depend on the problem at hand and on the input domain data distribution. In general, the transductive information, which corresponds to the knowledge of a node target, is valuable only if it allows to deduce something about neighboring nodes. Therefore, we expect that knowing that a node is part of the subgraph of interest (or not) is a valuable information if the subgraph nodes are isolated with respect to the rest of the graph. In order to verify such a hypothesis, we have generated different datasets, where graphs have different density levels, namely using an edge probability δ in $\{0.1, 0.5\}$. In sparse graphs, having small δs, the subgraph tends to be more isolated than in dense graphs.

Table III shows the results and confirms our intuition. Indeed, in the absence of noise, the difference between the performance of the mixed inductive–transductive GNN (50% column) and the pure inductive GNN (0% column) is larger for sparse graphs (with $\delta = 0.1$). Moreover, such a difference remains constant or increases with the injection of noise. In fact, the efficiency of the inductive learning decreases due to noise injection, while the transductive learning appears to be unaffected. In sparse graphs, where the information diffusion is particularly efficient, the difference between the learning frameworks may be evidenced by the noise level, whereas, in dense graphs, less sensible to transductive learning, the difference remains small.

2) Distance from the source: The performance achieved by the mixed inductive–transductive framework for this dataset is reported in Tables I-II. The distance from the source problem can be solved recursively, computing the distance of each node encountered in a path starting from a source node. In theory, such a recursive algorithm can be implemented in both the inductive and transductive framework. However, we found that inductive learning — for the chosen GNN and GCN configuration — suffers for the long–term dependency problem, since

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

the error signal has to be propagated, on long paths, from the target node to the source node. Such a problem is alleviated in the inductive–transductive model, where the error signal can be propagated also to the transductive nodes. In fact, the results show how the pure inductive models perform poorly, which is also confirmed by the observation that their performance is not influenced by the noise. Finally, the performance of the inductive–transductive model is boosted by the introduction of more and more transductive nodes.

3) Web Spam: The results achieved on the Web Spam benchmark, using both spam and non–spam hosts as transductive nodes, are shown in Tables I-II and Fig 2(b).

The performance increases when more transductive nodes are included, while it decreases in presence of increasingly noisy labels. Notice that, anyway, the loss in performance due to noise is slightly significant, suggesting that the node labels are not very informative in this case[20].

Interestingly, a well–known peculiarity of the Web is that spam pages tend to refer each other, while it is rare for a non–spam page to have hyperlinks to some spam pages [49]. More generally, we expect that spam pages are more clustered than non–spam pages. In order to understand how such a peculiarity can influence the proposed model, two different experiments were performed, in which the set of transductive nodes T has been chosen to collect only spam hosts or only non–spam hosts, respectively. The results confirmed such an asymmetry and show that the proposed model can take advantage of transductive spam hosts, while the advantage provided by the non–spam hosts is low (see Table IV).

4) Cora & Citeseer: The results achieved on Cora and Citeseer are collected in Tables I-II. Even if they cannot be directly compared with results reported in literature, due to the feature extraction procedure carried out on the labels and to the different splitting of the datasets, however they can be used to asses the proposed approach. They confirm that both the algorithms (i.e. GNN and GCN) can take advantage from the presence of transductive nodes. In particular, we can see that most of the times the highest scores for a certain level of noise are reached by the largest number of transductive nodes.

A final remark is due to the fact that the performance achieved by GCNs are lower than those obtained by GNNs on Cora e Citeseer and in most of the other experiments. It is worth mentioning, however, that here the comparison is largely unfair, since the used GNNs have much more parameters than the corresponding GCNs. The experiments have not been designed with the purpose of comparing those models, but for assessing their response to the inductive–transductive mixed learning.

5) ogb–Arxiv: Finally, the results for the ogb–arXiv dataset are shown in Tables V-VI. We used two different approaches to define the inductive and transductive nodes for this dataset. In the first approach, during testing, all the nodes in both the training and validation sets are used as transductive, whereas during training we use the same procedure as for the other datasets, where the nodes belonging to the training and validation sets are randomly split into transductive and inductive. We have named this approach ALL. The second approach is the one used also in the previous experiments, where even the test nodes are randomly split in transductive and inductive — this approach is called UNIFORM. Actually, the idea supporting the ALL approach is that of maximizing the information extracted from the training and validation sets, using all those nodes as transductive. This is also the most obvious way in which we can construct a learning environment for the proposed approach starting from an existing benchmark. Instead, the idea supporting UNIFORM is that of recreating, during the test phase, exactly the same distribution that exists for training. Formally, transductive nodes are removed from the test, to avoid to use them in the error function, which makes the results obtained with UNIFORM not comparable with other approaches.

The results also show that GCNs can take advantage of the transductive information in both the approaches, whereas the GraphSAGE model can exploit such an information only in the case of the UNIFORM approach. In general, the improvement is evident using the UNIFORM method, whereas it is very limited (if any) only for some percentages of transductive nodes in the case of the ALL method applied to GCNs. This is probably due to the importance of having similar distributions of the transductive and inductive nodes in training and test sets. Furthermore, the different placement of transductive nodes in UNIFORM and ALL can play an important role. Though this is not the goal of this paper, it suggests that performance can likely be further improved by optimizing the transductive and inductive node sets for both training and testing.

[20]Actually, in order to speed up the experiments, only a small subset of the available features from the original dataset have been used (10 out of 96).

IV. Conclusions

In this paper, a new mixed inductive–transductive learning framework has been proposed. A wide experimentation was carried out — based on the addition of noise on node labels and on the use of a variable percentage of transductive nodes — in order to disentangle the contributions of inductive and transductive learning, and to understand when one of the two frameworks is predominant over the other. These situations may arise from the data topology, e.g. graph with ”clustered” sets of nodes, or from the nature of the problem.

The proposed analysis served to highlight some interesting properties of the inductive–transductive learning algorithm for graphs. These properties, together with the experimental method adopted, could prove useful for the future conception of related models and algorithms.

It is a matter of future research to apply the proposed model, and the related experimental analysis, to other problems involving graph data, possibly coming from online social networks, mobile communications and also from the biomedical field. Furthermore, we will invest our future efforts in providing a theoretical basis to the clear insights gained from the experiments. Open questions include: to devise a theory that, on the base of some characteristics of the input graphs, allows to predict (and to explain) whether the inductive–trasductive approach will be useful; to establish what is the optimal balancing between transductive and inductive nodes and to discover how we can generate a training dataset, where

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

Type	Noise	Transductive Percentage				
		0%	*10%*	*25%*	*50%*	*Diff.* 50% - 0%
SPAM	*0%*	87.97 (1.3)	88.61 (1.2)	89 (2.08)	**93.11** (2.24)	5.14
	10%	86.56 (0.72)	88.54 (0.98)	90.37 (0.6)	**93.73** (0.98)	7.17
	25%	85.11 (1.36)	87.34 (1.05)	89.8 (0.77)	**93.47** (1.37)	8.36
	50%	83.74 (1.21)	84.11 (0.66)	87.51 (2.52)	**91.11** (1.47)	7.37
	100%	83.19 (1.31)	83.62 (1.04)	87.63 (1.08)	**90.33** (1.49)	7.14
NO SPAM	*0%*	88.18 (0.83)	84.54 (1.54)	**88.36** (1.96)	87.14 (4.88)	-1.04
	10%	87.86 (0.38)	85.56 (2.84)	89.18 (3.03)	**91.71** (3.29)	3.85
	25%	87.14 (0.88)	85.89 (0.53)	86.42 (2.75)	**91.10** (2.78)	3.96
	50%	85.16 (0.89)	83.82 (0.95)	88.12 (2.73)	**88.97** (5)	3.81
	100%	83.34 (1.38)	83.75 (1.46)	86.9 (1.05)	**90.9** (2.69)	7.56

TABLE IV: GNN accuracy results for the Web Spam dataset (the standard deviation is reported in brackets).

Method	Transductive Percentage			
	0%	*10%*	*25%*	*50%*
ALL	71.96 (0.31)	72.06 (0.29)	**72.37** (0.13)	71.99 (0.26)
UNIFORM	71.96 (0.31)	72.30 (0.29)	72.77 (0.15)	**73.14** (0.21)

TABLE V: GCN accuracy results for the ogb–arXiv dataset (the standard deviation is reported in brackets).

Method	Transductive Percentage			
	0%	*10%*	*25%*	*50%*
ALL	**71.71** (0.19)	69.94 (0.74)	70.47 (0.73)	71.08 (0.73)
UNIFORM	71.71 (0.19)	72.05 (0.31)	72.48 (0.32)	**72.85** (0.26)

TABLE VI: GraphSAGE accuracy results for the ogb–arXiv dataset (the standard deviation is reported in brackets).

the distribution of inductive and transductive nodes resemble that found in the test set.

REFERENCES

[1] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nature reviews genetics*, vol. 5, no. 2, p. 101, 2004.

[2] D. Arrell and A. Terzic, "Network systems biology for drug discovery," *Clinical Pharmacology & Therapeutics*, vol. 88, no. 1, pp. 120–125, 2010.

[3] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3, p. 186, 2009.

[4] T. Gärtner, J. Lloyd, and P. Flach, "Kernels and distances for structured data," *Machine Learning*, vol. 57, no. 3, pp. 205–232, 2004.

[5] R. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proceedings of the 19th International Conference on Machine Learning, ICML 2002*, C. Sammut and A. G. Hoffmann, Eds. Morgan Kaufmann Publishers Inc, 2002, pp. 315–322.

[6] J. Ramon and T. Gärtner, "Expressivity versus efficiency of graph kernels," in *First international workshop on mining graphs, trees and sequences*. Citeseer, 2003, pp. 65–74.

[7] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. Sep, pp. 2539–2561, 2011.

[8] F. Costa and K. D. Grave, "Fast neighborhood subgraph pairwise distance kernel," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress, 2010, pp. 255–262.

[9] F. Orsini, P. Frasconi, and L. De Raedt, "Graph invariant kernels," in *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 3756–3762.

[10] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of 18th International Conference on Machine Learning*, 2001.

[11] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning*. Mit Press, 2007.

[12] V. N. Vapnik, *Statistical Learning Theory*. Wiley, 1998.

[13] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[14] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE transactions on Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.

[15] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 714–735, 1997.

[16] M. Hagenbuchner, A. Sperduti, and A. Tsoi, "A self-organizing map for adaptive processing of structured data," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 491–505, 2003.

[17] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "Graph self-organizing maps for cyclic and unbounded graphs," *Neurocomputing*, vol. 72, no. 7-9, pp. 1419–1430, 2009.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[19] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[20] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[23] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[25] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[27] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[28] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehen-

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3054304, IEEE Transactions on Pattern Analysis and Machine Intelligence

sive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.
[29] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 290–297.
[30] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning." in *ICDM Workshops*, 2007, pp. 77–82.
[31] D. Zhou and C. J. Burges, "Spectral clustering and transductive learning with multiple views," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 1159–1166.
[32] W. Liu and S.-F. Chang, "Robust multi-class transductive learning with graphs," 2009.
[33] A. Belahcen, M. Bianchini, and F. Scarselli, "Web spam detection using transductive (inductive graph neural networks," in *Advances in Neural Networks: Computational and Theoretical Issues*. Springer, 2015, pp. 83–91.
[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "Computational capabilities of graph neural networks," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 81–102, 2008.
[35] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Seventh International Conference on Learning Representations*, 2018.
[36] M. Bianchini and M. Maggini, "Supervised neural network models for processing graphs," in *Handbook on Neural Information Processing*. Springer, 2013, pp. 67–96.
[37] A. Rossi, M. Tiezzi, G. M. Dimitri, M. Bianchini, M. Maggini, and F. Scarselli, "Inductive–transductive learning with graph neural networks," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer, 2018, pp. 201–212.
[38] F. Scarselli, A. C. Tsoi, and M. Hagenbuchner, "The vapnik–chervonenkis dimension of graph and recursive neural networks," *Neural Networks*, vol. 108, pp. 248–259, 2018.
[39] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[40] Y. Goldberg, *Neural Network Methods in Natural Language Processing*. Morgan & Claypool, 2017.
[41] D. Coppi, S. Calderara, and R. Cucchiara, "Transductive people tracking in unconstrained surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26(4), pp. 762–775, 2016.
[42] W. Liu and S. fu Chang, "S.f.: Robust multi-class transductive learning with graphs," in *In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR '09*, 2009, pp. 381–388.
[43] K. Li, N. Du, and A. Zhang, "Detecting ecg abnormalities via transductive transfer learning," in *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, ser. BCB '12. ACM, 2012, pp. 210–217.
[44] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International Conference on Machine Learning*, 2016, pp. 40–48.
[45] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations*, 2018.
[46] K. Wang, I. Shen, C. Huang, C.-H. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: when experts are not enough," *Quantitative Science Studies*, vol. 1, no. 1, pp. 396–413, February 2020.
[47] B. Bollobás and B. Béla, *Random graphs*. Cambridge university press, 2001, no. 73.
[48] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, S. Vigna *et al.*, "A reference collection for web spam," in *SIGIR forum*, vol. 40, no. 2, 2006, pp. 11–24.
[49] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri, "Know your neighbors: Web spam detection using the web topology," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 423–430.
[50] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

Giorgio Ciano received the B.E. degree in Information Engineering and the M.S. degree in Computer and Automation Engineering from University of Siena (Italy, 2018). Currently, is a PhD candidate in Smart Computing at the University of Florence (Italy). His primary research focuses on deep learning and computer vision.

Alberto Rossi received the B.E. degree in information engineering from University of Siena, Siena, Italy, in 2013, and the M.S. degree in computer and automation engineering from University of Siena, Italy, in 2017, with a score of 110 cum laude. He is working toward the PhD degree in smart computing at the University of Florence, Florence, Italy. His research interests include deep learning, data mining and computer vision.

Monica Bianchini received the Laurea degree (cum laude) in Applied Mathematics in 1989 and the PhD degree in Computer Science and Control Systems in 1995, from the University of Florence, Italy. She is currently an Associate Professor at the Department of Information Engineering and Mathematics of the University of Siena. Her research interests include machine learning, optimization, approximation theory, pattern recognition, and bioinformatics. She served/serves as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS (2003-09), Neurocomputing (2002-present), International Journal of Knowledge-Based and Intelligent Engineering Systems (2018) and has been the editor of numerous books and special issue in international journals on neural networks/structural pattern recognition. She is a permanent member of the editorial board of IJCNN, ICANN, ICPR, ANNPR, ICPRAM and KES.

Franco Scarselli Franco Scarselli received the Laurea degree with honours in computer science from the University of Pisa in 1989 and the PhD degree in computer science and automation engineering from the University of Florence in 1994. After the PhD he has been supported by foundations of private and public companies and by a postdoc of the University of Florence. In 1999, he moved to the University of Siena where he was initially a research associate and, currently, an associate professor at the department of Information Engineering. Franco Scarselli is author of more than fifty journal and conference papers and has been involved in several research projects, founded by public institutions and private companies, focused on software engineering, machine learning and information retrieval. Current theoretical research activity is mainly in the field of machine learning with a particular focus on adaptive processing of data structures, neural networks and approximation theory. Research interests include also image understanding, information retrieval and web applications.

0162-8828 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Carleton University. Downloaded on June 03,2021 at 21:01:23 UTC from IEEE Xplore. Restrictions apply.