Graph Kernel Neural Networks

Luca Cosmo, Giorgia Minello, Alessandro Bicciato, Michael M. Bronstein, Emanuele Rodolà, Luca Rossi, Andrea Torsello

***Abstract*—The convolution operator at the core of many modern neural architectures can effectively be seen as performing a dot product between an input matrix and a filter. While this is readily applicable to data such as images, which can be represented as regular grids in the Euclidean space, extending the convolution operator to work on graphs proves more challenging, due to their irregular structure. In this paper, we propose to use graph kernels, *i.e.*, kernel functions that compute an inner product on graphs, to extend the standard convolution operator to the graph domain. This allows us to define an entirely structural model that does not require computing the embedding of the input graph. Our architecture allows to plug-in any type of graph kernels and has the added benefit of providing some interpretability in terms of the structural masks that are learned during the training process, similarly to what happens for convolutional masks in traditional convolutional neural networks. We perform an extensive ablation study to investigate the model hyper-parameters' impact and show that our model achieves competitive performance on standard graph classification and regression datasets.**

***Index Terms*—Graph neural network, graph kernel, deep learning**

I. Introduction

In recent years, graph neural networks (GNNs) have gained increasing traction in the machine learning community. Graphs have long been used as a powerful abstraction for a wide variety of real-world data where structure plays a key role [1], from collaborations [2], [3] to biological [4], [5] and physical [6] data, to mention a few. Before the advent of GNNs, *graph kernels* provided a principled way to deal with graph data in the traditional machine learning setting [7], [8], [9]. However, with the attention of machine learning researchers steadily shifting away from hand-crafted features toward end-to-end models where both the features and the model are learned together, neural networks have quickly overtaken kernels as the framework of choice to deal with graph data, leading to multiple popular architectures such as [3], [4], [10].

The main obstacle that both traditional and deep learning methods have to overcome when dealing with graph data is also the source of interest for using graphs as data representations. The richness of graphs means that there is no obvious way to embed them into a vector space, a necessary step when the learning method expects vectors in input. One reason is the lack of a canonical ordering of the nodes in a graph, requiring either permutation-invariant operations or an alignment to a reference structure. Moreover, even if the order or correspondence can be established, the dimension of the embedding space may vary due to structural modifications, *i.e.*, changes in the number of nodes and edges.

In traditional machine learning, kernel methods, and particularly graph kernels, provide an elegant way to sidestep this issue by replacing explicit vector representations of the data points with a positive semi-definite matrix of their inner products. Thus, any algorithm that can be formulated in terms of scalar products between input vectors can be applied to a set of data (such as graphs) on which a kernel is defined.

GNNs apply a form of generalized convolution operation to graphs, which can be seen as a message-passing strategy where the node features are propagated over the graph to capture node interactions. Compared to standard convolutional neural networks (CNNs), the main drawback of these architectures is the lack of a straightforward way to interpret the dependency between the output and the presence of certain features and structural patterns in the input graph. In this paper, we propose a neural architecture that bridges the two worlds of graph kernels and GNNs. The key intuition underpinning our work is that there exists an analogy between the traditional convolution operator, which can be seen as performing an inner product between an input matrix and a filter, and graph kernels, which compute an inner product of graphs. As such, graph kernels provide the natural tool to generalize the concept of convolution to the graph domain, which we term *graph kernel convolution* (GKC). Given a graph kernel of choice, in each GKC layer, the input graph is compared against a series of structural masks (analogous to the convolutional masks in CNNs), which effectively represent learnable subgraphs. These, in turn, can offer better interpretability in the form of insights into what structural patterns in the input graphs are related to the corresponding output (*e.g.*, the carcinogenicity of a chemical compound).

Our main contributions are:

- We introduce a neural model that is fully structural, unlike existing approaches that require embedding the input graph into a larger, relaxed space. While the latter allows one to seek more complex and arbitrary decision boundaries, it also has the potential of overfitting the problem and creating more local optima, as evidenced by the need to use dropout strategies to avoid overfitting in Graph Convolutional Networks (GCNs) [11], [12].
- In order to cope with the non-differentiable nature of graphs, we let the learned structural masks encode distri-

L. Cosmo, G. Minello, A. Bicciato, and A. Torsello are with Ca' Foscari University of Venice, Venice, Italy. M. M. Bronstein is with Oxford University, United Kingdom. E. Rodolà is with Sapienza University of Rome, Rome, Italy. L. Rossi is with The Hong Kong Polytechnic University, Hong Kong. Corresponding author: luca.rossi@polyu.edu.hk

butions over graphs, which in turn allows us to compute the gradient of the expected kernel response between the learned masks and the input subgraphs. This is, in turn, achieved through a computationally efficient importance sampling approach.
- Our architecture allows to plug-in any type of graph kernel (not just differentiable kernels as in [13], [14]).
- As an added benefit, our model provides some interpretability regarding the structural masks learned during the training process, similarly to convolutional masks in traditional CNNs.
- Finally, we analyze the expressive power of our model, and we argue that it is greater than that of standard message-passing GNNs and the equivalent Weisfeiler–Lehman (WL) graph isomorphism test.

The fundamental goal of our network is to reconstruct the structural information needed for classification and is thus particularly suited for problems where the structure plays a pivotal role. These are often graph classification and regression problems, where structure provides the most relevant information for classification, while node/edge features take a secondary role. The situation is usually reversed for node classification problems, where the distribution of features across the immediate neighbors of a node holds most of the information needed to classify the node itself. For these reasons, in this paper, we focus on graph classification and regression problems. However, we stress that our model can indeed be extended to tackle node-level tasks.

The remainder of this paper is organized as follows. In Section II, we review the related work. In Section III, we present our model, where graph kernels are used to redefine convolution on graph data. In Section IV, we investigate the hyper-parameters of our architecture through an extensive ablation study, provide some insight into the interpretability of the structural masks, and evaluate our architecture on standard graph classification and regression benchmarks. Finally, Section V concludes the paper.

II. Related work

The majority of graph kernels belong to one of two main categories: 1) bag-of-structures and 2) information propagation kernels. Bag-of-structures kernels compute the similarity between a pair of input graphs by first decomposing them into simpler substructures and then counting the number of isomorphic substructures between the two input graphs. Depending on the type of substructure considered, one can build a multitude of different kernels, *e.g*., subtrees [15], shortest paths [16], and graphlets [17]. Information propagation kernels, on the other hand, include methods where pairs of input graphs are compared based on how information diffuses on them. Examples include random walk kernels [18], [19], quantum walk kernels [8], [20], [21], and kernels based on iterative label refinements [7]. While some kernels work only on undirected and unattributed graphs, other kernels are designed to handle attributes as well, either discrete- or continuous-valued [7], [22]. For a detailed review and historical perspective on graph kernels, we refer the reader to the recent survey of Kriege *et al.* [23].

In recent years, with the advent of deep learning and the renewed interest in neural architectures, the focus of graph-based machine learning researchers has quickly moved to extending deep learning approaches to deal with graph data. Fundamentally, the principle underpinning most GNNs is that of exploiting the structure of the graph to propagate the node feature information iteratively. One of the first papers to propose the idea of GNNs is that of Scarselli *et al.* [24], where an information diffusion mechanism is used to learn the nodes' latent representations by exchanging neighborhood information. Sperduti and Starita [25] and Micheli [26] had previously relied instead on recursive operators, which can be seen as a variation of convolutional operators where different layers share the same weights, to adapt neural networks to operate on structured data such as graphs. Most modern GNNs, however, appear to fall within three main categories, as discussed by Bronstein *et al.* [27]: 1) convolutional [3], [28], [29], [30], [31], 2) attentional [10], and 3) message passing [4]. The message-passing model is often seen as the most general one and has been shown to be formally equivalent to the WL graph isomorphism test under some technical conditions [32], [33].

An increasing number of researchers are trying to overcome the latter limitation by moving away from standard message-passing GNNs [34], [35]. Eliasof *et al.* [36] propose a reinterpretation of graph convolution in terms of partial differential equations on graphs, noting that different problems can benefit from different networks dynamics, hence suggesting the need to look beyond diffusion. Bevilaqua *et al.* [37] represent graphs as bags of substructures and show that this allows the WL test to distinguish between otherwise indistinguishable graphs. Bouritsas *et al.* [38] take instead an alternative approach where augmenting the graph nodes attributes with positional encoding is shown to improve the ability of the WL test to discriminate between non-isomorphic graphs. The field of GNNs is, in general, in rapid and continuous expansion, so presenting a comprehensive review of the literature is beyond the scope of this paper. Instead, we refer the interested readers to [39], [27].

In this work, we argue that a natural extension of the convolution operation to the graph domain, and thus an ideal candidate to build graph CNNs, already exists in the form of graph kernels. A number of works in the literature have investigated potential synergies between GNNs and kernels. Lei *et al.* [40] introduce a class of deep recurrent neural architectures to show that this lies in the reproducing kernel Hilbert space (RKHS) of graph kernels. Nikolentzos *et al.* [41] compute continuous embeddings of graphs using kernels and plug them into a neural network. Xu *et al.* [32] and Morris *et al.* [33] show that GNNs have the same expressiveness as the WL graph kernel [7] and propose a new generalized architecture with increased expressive power. Du *et al.* [42] take a somewhat different approach and exploit neural networks to introduce a new graph kernel. This, in turn, is shown to be equivalent to an infinitely wide GNN initialized with random weights and trained with gradient descent. Chen *et al.* [43] propose a graph neural architecture where each layer enumerates local substructures around each node and then maps them to an RKHS via a Gaussian kernel mapping. In the context of graph compression, Bouritsas *et al.* [44] learn how to best decompose the graph into small substructures that are also learned. Navarin *et al.* [45] propose a modification of DGCNN [46], a popular GNN architecture, where a multi-task learning approach is used to drive the learned node embeddings to be close to those computed by graph kernels.

III. GNNs from Graph Kernels

In this section we introduce the proposed neural architecture, the Graph Kernel Neural Network (GKNN). Before that, however, we introduce some background concepts that will help better understand the proposed model.

A. Preliminaries

Kernel methods. This class of algorithms is capable of learning when presented with a particular positive pairwise measure on the input data, known as a kernel. Consider a set X and a positive semi-definite kernel function $\mathcal{K} : X \times X \to \mathbb{R}$ such that there exists a map $\phi : X \to H$ into a Hilbert space H and $\mathcal{K}(x, y) = \phi(x)^\top \phi(y)$ for all $x, y \in X$. Crucially, X can represent any set of data on which a kernel can be defined, from $\mathbb{R}^d$ to a finite set of graphs. Hence, the field of machine learning is ripe with examples of graph kernels (see Section II), which are nothing but positive semi-definite pairwise similarity measures[1] on graphs. These can be either implicit (only $\mathcal{K}$ is computed) or explicit (ϕ is also computed).

Weisfeiler-Lehman test. The WL kernel [7] is one of the most powerful and commonly used graph kernels, and it is based on the 1-dimensional Weisfeiler-Lehman (1-WL) graph isomorphism test. The idea underpinning the test is to partition the node set by iteratively propagating the node labels between adjacent nodes. With each iteration, the set of labels accumulated at each node of the graph is then mapped to a new label through a hash function. This procedure is repeated until the cardinality of the set of labels stops growing. Two graphs can then be compared in terms of their label sets at convergence, with two graphs being isomorphic only if their label sets coincide.

Learning on graphs. Let $\mathcal{G} = (V, X, E)$ be an undirected graph with $|V|$ nodes and $|E|$ edges, where each node v is associated to a label $x(v)$ belonging to a dictionary $\mathbf{D}$. Note that, generally, graphs can be directed and have continuous and discrete attributes on both the nodes and edges; however, for simplicity in this paper, we restrict our attention to undirected node-labeled graphs. Indeed, the model described in this paper can be easily applied to directed, edge-labeled graphs as well. A common goal in graph machine learning problems is to produce a vector representation of $\mathcal{G}$ that is aware of both the node labels and the structural information of $\mathcal{G}$. Generalizing the convolution operator to graphs, GNNs learn the parameters Θ of a function h that performs message passing [4] in the 1-hop neighborhood of each node $v \in V$,

$$z(v) = \sum_{u \in \mathcal{N}^1_{\mathcal{G}}(v)} h_\Theta \left(x(v), x(u) \right), \tag{1}$$

[1]Note that while several kernels can be interpreted as similarity measures, this is not always the case [47].

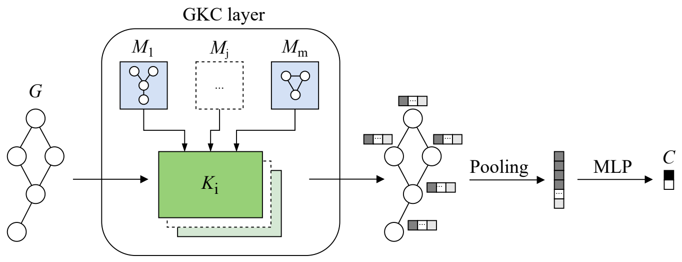


Fig. 1. The proposed GKNN architecture. The input graph is fed into one or more GKC layers, where subgraphs centered at each node are compared to a series of structural masks through a kernel function. The output is a new set of real-valued feature vectors associated to the graph nodes. We obtain a graph-level feature vector through pooling on the nodes features, which is then fed to an MLP to output the final classification label.

where $\mathcal{N}^1_{\mathcal{G}}(v)$ denotes the 1-hop neighborhood of v. The convolution is usually repeated for L layers, and the final vectorial representation Z_{pool} is obtained by applying a node-wise permutation invariant aggregation operator $\square$ on the node features $z(v)$.

B. Proposed architecture

The core feature of the proposed model is the definition of a new approach to perform the convolution operation on graphs. Unlike the diffusion process performed by message passing techniques, we design the *Graph Kernel Convolution* (GKC) operation in terms of the inner product between graphs computed through a graph kernel function.

Given a graph $\mathcal{G} = (V, X, E)$ with $|V|$ nodes, we extract $|V|$ subgraphs $\mathcal{N}_{\mathcal{G}}{}^r(v)$ of radius r, centered at each vertex $v \in V$. Each subgraph consists of the central node v, the nodes at distance at most r from v, and the edges between them. Each such subgraph is then compared to a set of learnable structural masks $\{\mathcal{M}_1, \dots, \mathcal{M}_m\}$ through

$$z_i(v) = \mathcal{K}(\mathcal{M}_i, \mathcal{N}^r_{\mathcal{G}}(v)), \tag{2}$$

where the resulting feature vector $z(v)$ is a non-negative real-valued m-dimensional vector collecting the kernel responses, and the i-th structural mask $\mathcal{M}_i = (V_{\mathcal{M}_i}, X_{\mathcal{M}_i}, E_{\mathcal{M}_i})$ is a first-order stochastic graph, *i.e.*, a distribution over graphs with node set $V_{\mathcal{M}_i}$ where the edges $e \in E_{\mathcal{M}_i}$ are sampled from independent Bernoulli trials each with its own parameter θ_e and each node label $x \in X_{\mathcal{M}_i}$ is sampled from the corresponding discrete probability distribution over a dictionary $\mathbf{D}$ with parameters $\phi_{xd} \forall d \in \mathbf{D}$ each expressing the probability of x assuming value d. The probability distributions θ and ϕ over edges and node labels defining each mask are effectively the learnable parameters of the GKC layer.

Multi-layer architecture. The output node feature $z(v) \in \mathbb{R}^m_+$ of a GKC operation consists of a continuous vector containing the graph kernel responses between the structural masks and the subgraph centered on the node v. For graph kernels functions requiring discrete labels, the output feature vector thus needs to be discretized before giving it as input to the next layer. In this case, we interpret $z(v)$ as an unnormalized probability distribution over a set of m labels from which we sample the new label $x(v)$ for the node v at the next GKC layer.

Since in this paper we focus on graph-level tasks, the final graph embedding after L layers is obtained by $\square(Z^1|\dots|Z^L)$, where Z^ℓ are the output features of the ℓth layer and $\cdot|\cdot$ indicates the concatenation of node-wise features. This is then fed to a multilayer perceptron (MLP) layer to obtain the final classification. In particular, in all our experiments we use sum pooling as the aggregation operator $\square$. Figure 1 shows an overview of the proposed multilayer architecture.

Cross-entropy loss. Our learning problem can be formulated as follows: given a set $\{\mathcal{G}_1,\dots,\mathcal{G}_B\}$ of B training input graphs with node labels belonging to the dictionary $\mathbf{D}$ and associated class labels $y_1\dots y_B$, our goal is to find the optimal parameters Φ and the masks $\mathcal{M}_i$ with corresponding edge observation probabilities Θ of the model g that minimize the cross-entropy loss

$$loss_{CE} = \sum_{i=1}^{B} \textit{cross-entropy}(\mathbb{E}_\Theta[g_{\mathcal{M}_1\dots\mathcal{M}_m,\Phi}(\mathcal{G}_i)], y_i)\,. \quad (3)$$

Jensen–Shannon divergence loss. Depending on the number of structural masks to learn, we experimentally observed that different structural masks can give a similar response over the same node. To mitigate this behavior and push the model to learn a more descriptive node feature vector, we propose to regularize the structural masks learning process by adding a Jensen–Shannon divergence (JSD) loss. The JSD is computed between the feature dimensions considered as probability distributions over the nodes of the graph.

Let $P_i = \{\alpha z_i(v)|v \in V\}$ be the probability distribution induced by the i-th mask over the graph nodes. α is a scaling factor ensuring that $\sum_v P_i(v) = 1$. We define the JSD loss as

$$loss_{JSD} = -H\left(\sum_{i=1}^{m} P_i\right) + \sum_{i=1}^{m} H\left(P_i\right)\,, \quad (4)$$

where $H(P)$ is the Shannon entropy. The final loss we optimize is the sum of 1) the cross-entropy loss of Eq. 3 2) $loss_{JSD}$ multiplied by a weighting factor λ, i.e.,

$$loss = loss_{CE} + \lambda loss_{JSD}\,. \quad (5)$$

Note that while previous work in the literature used the JSD to generalize the widely used cross-entropy loss function [48], here we use it as a regularization term. That is, rather than minimizing the JSD or cross-entropy in order to limit the distance to a target class [49], we maximize the JSD to force the learned distributions to be as far away from one another as possible.

C. Optimization strategy

When optimizing the structural masks, we need to consider two main challenges. First, the number of nodes of the subgraphs is not fixed and can, in principle, vary from 1 to n. Second, since graph kernel functions are not in general differentiable, the automatic differentiation mechanism of common neural network optimization libraries cannot be directly applied to our model.

Structural masks representation. When defining the space of graphs on which we want to optimize the structural masks $\mathcal{M}$, we have to consider the possible substructures present in the input graph that characterize it as belonging to a specific class. Assuming that this knowledge is not known *a priori*, we should allow to learn structures as large as the graph itself. Unfortunately, since the space of graphs grows exponentially with the number of nodes, this would be impractical. Moreover, under the assumption of the presence of localized characterizing substructures, we usually need graphs of few nodes to capture their presence. In our implementation, we fix a maximum number of nodes p_{max} for each substructure and optimize for structural masks in the space

$$\mathcal{M} \in \bigcup_{p=1}^{p_{max}} \left(\tilde{\mathbf{G}}_p \times \mathbf{D}^p\right)\,, \quad (6)$$

where $\tilde{\mathbf{G}}_p$ indicates the set of all possible connected graphs of p nodes, and $\mathbf{D}^p$ the labeling space of p nodes. The impact of this hyper-parameter is studied in the ablation study in Section IV. With a slight abuse in notation, here $\mathcal{M}$ represents a discrete graph with discrete node labels, but we will relax it to its stochastic version in the subsequent paragraph.

Structural masks optimization. Graph kernels, as functions operating on discrete graph structures, are in general not differentiable. In order to be able to optimize the structural masks, we relax them to be distributions over graphs and take the expectation of the kernel over said distributions. Namely, let $\mathcal{M}_i = (V_{\mathcal{M}_i}, X_{\mathcal{M}_i}, E_{\mathcal{M}_i})$ be our *i*-th structural mask, where $V_{\mathcal{M}_i}$ is the set of p nodes, $X_{\mathcal{M}_i}$ is the set of p discrete probability distributions over the $|D|$ labels, and $E_{\mathcal{M}_i}$ is the edge observation models consisting of $\binom{n}{2}$ independent Bernoulli trials with (learnable) parameters $\theta_1,\dots,\theta_{\binom{n}{2}}$.

A sample $G \sim \mathcal{M}_i$ is a graph $G = (V, X, E)$ whose edge set E is the result of sampling from the Bernoulli trials to decide whether the corresponding edge is present, and labels X are individually sampled from the corresponding node label distributions. With this model to hand, we take as the kernel response between the subgraph $\mathcal{N}_\mathcal{G}^\tau(v)$ and the mask $\mathcal{M}_i$ the expectation of the kernel over the distribution of graphs associated with $\mathcal{M}_i$, *i.e.*,

$$z_i(v) = \mathbb{E}_{G\sim\mathcal{M}_i}[\mathcal{K}(G, \mathcal{N}_\mathcal{G}^\tau(v))]\,. \quad (7)$$

The gradient is then taken with respect to the edge observation parameters Θ.

Clearly, computing the full expectation for the kernel response and its gradient is computationally too demanding. For this reason, in the next section we discuss a sampling strategy for both.

D. Expected kernel estimation through importance sampling

In order to efficiently compute the kernel responses and their gradient with respect to the edge observation parameters Θ we adopt an importance sampling strategy. Importance sampling is a Monte Carlo method for evaluating the expectation of a function over a particular distribution while only having samples generated from a different distribution than the distribution of interest. More formally, assume you have a function

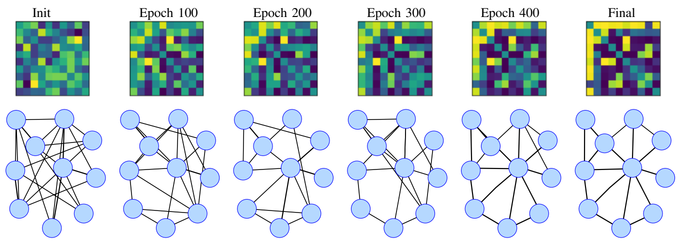


Fig. 2. Example of model training. Top: the matrix of edge observation probabilities associated to the mask $\mathcal{M}_i$ for successive epochs. Bottom: the corresponding mode graphs.

$f : X- > \mathbb{R}$ over a domain X and two distributions p and g over such domain. Then we have

$$\mathbb{E}_p[f] = \int_X f(x)\, p(x)dx = \int_X f(x)\frac{p(x)}{g(x)}\, g(x)dx = \mathbb{E}_g\left[f(x)\frac{p(x)}{g(x)}\right] . \quad (8)$$

Eq. 8 suggests that the expectation of f with respect to the distribution p can be obtained from the expectation with respect to the distribution g by adding the correction term $\frac{f(x)}{g(x)}$. Importance sampling is often used to reduce the sampling variance. Indeed, if $g(x) = \frac{1}{\mathbb{E}_p[f]} f(x)p(x)$, then a single sample is sufficient to estimate $\mathbb{E}_p[f]$. More in general, if the samples are concentrated where both the response f and the density p are high, then the sampling variance is greatly reduced. To this end, and observing that our first-order random graph model associated with each mask $\mathcal{M}_i$ is strongly unimodal being the combination of independent Bernoulli trials, we concentrate our sampling around the mode of the distribution. In particular, we sample from the distribution of graphs associated with the mask $\mathcal{M}_i$ the mode G_M and the graphs that can be obtained from the mode with a single edge edit operation, *i.e.*, the neighboring graphs. To this end, recall that in our setting the mode is the graph such that edge e is present if and only if $\theta_e > 0.5$. Letting $p_e = \max(\theta_e, 1 - \theta_e)$, we have that p_e is the probability that edge e is sampled as present or absent as in the mode G_M and thus the probability of sampling G_M is

$$P(G_M) = \prod_e p_e \quad (9)$$

and the density of any neighbouring graph G_e obtained by editing the edge e is

$$P(G_e) = P(G_M)\frac{1 - p_e}{p_e} . \quad (10)$$

Assuming that we are sampling uniformly from the set consisting of the mode and its neighboring graphs, our sampling density is $\frac{1}{\binom{n}{2}+1}$ over that set, 0 otherwise.

With this setup, we can estimate the expected kernel response as

$$z_i(v) = \mathbb{E}_{G\sim\mathcal{M}_i}[\mathcal{K}(G, \mathcal{N}^r_{\mathcal{G}}(v))] \approx \frac{\binom{n}{2}+1}{|\{G\}|}\sum_{G\sim\mathcal{M}_i} P(G)\mathcal{K}(G, \mathcal{N}^r_{\mathcal{G}}(v)) = \frac{P(G_M)\left(\binom{n}{2}+1\right)}{|\{G\}|}\Bigg(\mathcal{K}(G_M, \mathcal{N}^r_{\mathcal{G}}(v)) + \sum_e \mathcal{K}(G_e, \mathcal{N}^r_{\mathcal{G}}(v))\frac{1 - p_e}{p_e}\Bigg) \quad (11)$$

where $\{G\}$ is the set of samples, $P(G)$ is the sampling probability, G_M is the mode graph, and the summation over e refers to the summation over the sampled neighbouring graphs G_e, where e is the edge that differs from the mode. Note that the first term is due to the mode. If the mode is not among the samples that term can be dropped.

Similarly, we differentiate the expectation as follows:

$$\frac{\partial z_i(v)}{\partial\theta_{e'}} \approx \frac{\left(\binom{n}{2}+1\right)}{|\{G\}|}\Bigg(\mathcal{K}(G_M, \mathcal{N}^r_{\mathcal{G}}(v))\frac{P(G_M)}{p_{e'}}\frac{\partial p_{e'}}{\partial\theta_{e'}} + \sum_{e\neq e'}\mathcal{K}(G_e, \mathcal{N}^r_{\mathcal{G}}(v))\frac{P(G_e)}{p_{e'}}\frac{\partial p_{e'}}{\partial\theta_{e'}} + \mathcal{K}(G_{e'}, \mathcal{N}^r_{\mathcal{G}}(v))\frac{P(x_{e'})}{1 - p_{e'}}\frac{\partial(1 - p_{e'})}{\partial\theta_{e'}}\Bigg) . \quad (12)$$

Here too, we have a term associated with the mode, plus one associated with the neighbor obtained by editing the edge e'. Either term can be dropped if the corresponding graph is not among the samples.

Taking out the factor $\frac{P(G_M)}{p_{e'}}\frac{\partial p_{e'}}{\partial\theta_{e'}}$, we obtain

$$\frac{\partial z_i(v)}{\partial\theta_{e'}} \approx \frac{P(G_M)}{p_{e'}}\frac{\left(\binom{n}{2}+1\right)}{|\{G\}|}\frac{\partial p_{e'}}{\partial\theta_{e'}}\Bigg(\mathcal{K}(G_M, \mathcal{N}^r_{\mathcal{G}}(v)) + \sum_{e\neq e'}\mathcal{K}(G_e, \mathcal{N}^r_{\mathcal{G}}(v))\frac{1 - p_e}{p_e} - \mathcal{K}(G_{e'}, \mathcal{N}^r_{\mathcal{G}}(v))\Bigg) . \quad (13)$$

Finally, we note that $\frac{\partial p_{e'}}{\partial\theta_{e'}} = \pm 1$, with the sign being positive when the edge is observed in the mode, negative when it is not.

Recall that, in our setup, each node of the graphs and masks has an associated distribution over node labels. During inference, we sample from these distributions to obtain the labeled graphs to pass to the kernel. During the learning phase, we compute the gradient of the output $z_i(v)$ with respect to the feature distribution of the mask by differentiating its expectation with a similar Monte Carlo approach. However, in this case, the expectation is taken not only over the structures of the graphs sampled from the mask, but also over the labels sampled from the corresponding distribution ℓ associated with each node in the mask

$$z_i(v) = \mathbb{E}_{G\sim\mathcal{M}_i,\, l\in\ell^{\otimes n}}[\mathcal{K}(G^l, \mathcal{N}^r_{\mathcal{G}}(v))] \quad (14)$$

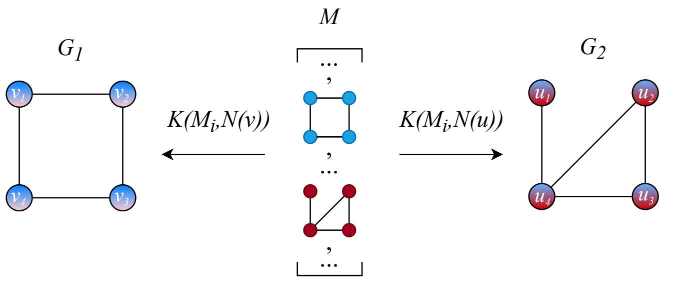


Fig. 3. Assuming the set of masks M spans the space of graphs over n nodes, if the radius of each $N^r_{\mathcal{G}_1}(v)$ is such that it spans the entire graph, the responses on the nodes of $\mathcal{G}_1$ will be identical and will peak for any mask fractionally equivalent to $\mathcal{G}_1$. Here, we use blue and red to highlight the masks with the highest response on $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively.

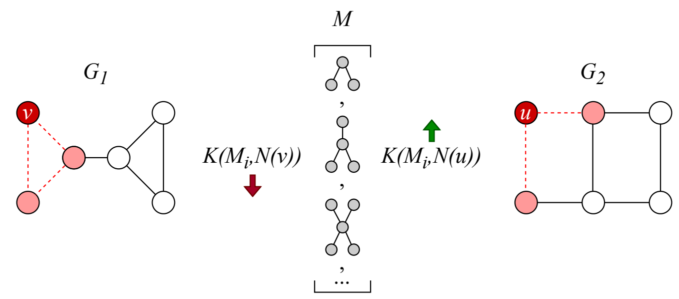


Fig. 4. $\mathcal{G}_1$ and $\mathcal{G}_2$ cannot be distinguished by the 1-WL test, however the GKNN correctly identifies them as non-isomorphic graphs. If M is the set of all structural masks representing stars up to $n-1$ nodes, no star mask will achieve maximum similarity on the radius 1 subgraphs centered on the nodes of $\mathcal{G}_1$, while the same maximal response will be achieved when considering the radius 1 subgraphs centered on the nodes of $\mathcal{G}_2$.

where G^l is the graph with attached node labels l and $\ell^{\otimes n}$ is the concatenated label distribution over all the nodes of the mask.

Assuming again that the distribution over the structure is concentrated around the mode G_M, we can then estimate the gradient with respect to the label distribution $\ell^{\otimes n}$

$$\nabla_{\ell^{\otimes n}} z_i(v) \approx \frac{1}{|\{l\}|}\sum_{l\in\ell^{\otimes n}} (l-\bar{l})\Big(\mathcal{K}(G^l_M, \mathcal{N}^r_{\mathcal{G}}(v)) - \mathcal{K}(G^{\bar{l}}_M, \mathcal{N}^r_{\mathcal{G}}(v))\Big) \tag{15}$$

where $\bar{l}$ is the label set sampled in the forward pass, and l and $\bar{l}$ taken as vectors are the concatenated one-hot encodings of the sampled labels.

Figure 2 shows the matrix of edge observation probabilities associated to a sample mask $\mathcal{M}_i$ for successive training epochs (top) and the corresponding mode graphs (bottom). Note that as the epoch number increases the probability distribution converges to the mode. This allows us to consider only the mode during inference, which results in a faster and deterministic prediction, *i.e.*, we do not need to estimate the expected value of the output of our model. Note that here, as well as in the experiments described in Section IV, we initialized the edge observation probabilities with random samples from a uniform distribution. Alternatively, one could initialize these probabilities to reflect the presence of frequent subgraphs mined from the training data or predefined patterns of interest, such as stars and cycles. We decide instead to limit the bias we introduce in the model, with an additional motivation being the fact that initializing the probabilities in this way appears to give the network sufficient flexibility to learn masks that resemble discriminative patterns found in the input graphs (see Section IV for further details).

E. Expressive power

In [33], Morris *et al.* show that the expressive power of standard GNNs (*i.e.*, GNNs that only consider the immediate neighbors of a node when updating the labels) is equivalent to that of the 1-WL test. In this subsection, we discuss our model's expressive power and argue that this is higher than the 1-WL test and thus standard message-passing GNNs. For this to be the case, our model should be both able to distinguish every pair of graphs that can be distinguished by the 1-WL test and it should be able to distinguish between pairs of graphs where the 1-WL test fails.

Graphs distinguished by the 1-WL test. Consider two unlabelled graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ with the same number of nodes $n = |V_1| = |V_2|$, such that the 1-WL test can distinguish between them. Then we can also build a GKNN able to distinguish between them by adopting the WL kernel in the GKC layer, and the radius r is such that the subgraphs $N^r_{\mathcal{G}_1}(v)$ span the entire graph $\mathcal{G}_1$ (similarly for $\mathcal{G}_2$). In fact, the set of labels computed against any one mask will be the same for every node in graph $\mathcal{G}_1$ ($\mathcal{G}_2$) since all the subgraphs are identically equal to the graph itself. Assuming that we have a set of structural masks $\{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$ that span the space of graphs over n nodes, then, under the hypothesis that the 1-WL test can distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$, the set of labels computed by the GKC layer (*i.e.*, the responses of the masks) while identical on all the nodes of each graph, will differ between the two graphs (see Figure 3). In particular, the response will peak for any mask fractionally equivalent to the graph itself [50]. Therefore, under any reasonable pooling strategy, the GKNN will be able to distinguish between the two graphs.

We can easily see that the same holds for an arbitrary subgraph radius r under the assumption of a sufficient number of layers. For simplicity, we will assume $r = 1$, but, as shown in the previous argument, increasing r increases the descriptive power of the GKC layer. Note that the full GKC Layer can be seen as 1) a label propagation step followed by 2) a hashing step. The label propagation is given by the computation of the WL kernel against a structural mask, due to the 1-WL iterations on the subgraphs.

The hashing, on the other hand, comes from the implicit dot product against the structural masks. This is, in essence, a projection onto the space spanned by them and forces a hashing where the adopted masks form a representation space for the hashing itself. Note that, under the assumption of a sufficiently large (possibly complete) set of structural masks $\{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$ spanning the space of graphs over n nodes,

TABLE I
BEST MODELS AVERAGE TRAINING TIME PER EPOCH (OVER FOLDS) AND INFERENCE TIME FOR THE DATASETS CONSIDERED IN THE PRESENT WORK.

	Per epoch training time (s)				Per sample inference time (ms)			
	Ours (WL)	GIN	DiffPool	ECC	Ours (WL)	GIN	DiffPool	ECC
PROTEINS	111.92 ± 59.85	1.55 ± 0.14	5.45 ± 0.20	2.68 ± 0.21	14.73 ± 5.48	4.09 ± 0.94	13.71 ± 0.16	9.02 ± 1.29
MUTAG	10.01 ± 1.92	0.21 ± 0.04	0.70 ± 0.03	0.32 ± 0.05	16.66 ± 1.12	3.29 ± 1.18	9.25 ± 0.18	7.28 ± 1.03
PTC	9.30 ± 3.67	0.40 ± 0.07	1.49 ± 0.10	0.59 ± 0.05	11.76 ± 4.68	3.24 ± 1.22	9.38 ± 0.19	8.18 ± 0.67
NCI1	83.45 ± 1.80	5.86 ± 0.72	17.90 ± 0.52	9.07 ± 0.44	8.02 ± 0.36	4.43 ± 0.69	9.35 ± 0.16	8.91 ± 1.71
IMDB	36.80 ± 3.59	1.44 ± 0.22	4.25 ± 0.15	1.88 ± 0.12	8.90 ± 0.16	4.23 ± 0.72	9.25 ± 0.16	7.64 ± 1.15
ZINC	225.16	6.21	21.33	12.45	2.16	4.33	5.92	4.80

the dot product induced hashing will not introduce conflicts between nodes with different labels. In other words, distinct labels under the 1-WL iterations will eventually be mapped to distinct labels at some GKC layer. Note, however, that under these assumptions the expressive power is greater or equal than that of 1-WL and labels that are equivalent under 1-WL iteration can indeed be mapped to distinct labels by some GKC layer. In fact, the propagation for 1-WL uses only the star of radius 1 around a particular node, while our scheme uses the full 1-hop subgraph, which includes connections between neighbors. Therefore, given a sufficient number of GKC layers and a reasonable pooling strategy, the GKNN will be able to distinguish between the two input graphs.

Graphs where the 1-WL test fails. Not only the GKNN can distinguish between pairs of graphs that are correctly distinguished by the 1-WL test, but it also succeeds where the 1-WL test fails. To see that this is the case, consider the standard example of two graphs with 6 nodes, where $\mathcal{G}_1$ is the dumbbell graph obtained by connecting two 3-nodes cycles with one edge, while $\mathcal{G}_2$ is a 2×3 lattice graph (see Figure 4). Both graphs have the same degree distribution, with 4 nodes of degree 2 and 2 nodes of degree 3. As explained in [51], [52], despite being non-isomorphic, the two graphs are considered identical by the 1-WL test (in other words, the WL relabeling procedure converges to the same set of labels for the two graphs). To see why this happens, consider that through the propagation phases, each node is only made aware of what degree other nodes in the graph have, and not how many such nodes can be reached. However, we can easily see that this is not an issue for the GKNN.

Assume we have a GKC layer with radius $r = 1$ that adopts a WL kernel and a set of structural masks $M = \{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$ representing all stars up to $n-1$ nodes, where n denotes the number of nodes of the input graph. The presence of a triangle in the graph induces the existence of at least three subgraphs of radius 1 with connections between neighbors, *i.e.*, not stars. On these subgraphs, no star masks will achieve maximum similarity under the WL kernel. Having a pooling process that normalizes the masks responses with respect to size and maximizes over the masks on the same node, implies that each node will have the same maximal response if it does not contain triangles and a lower response otherwise. A min-pooling process over the node features will then be able to discriminate between graphs with and without triangles. Now consider the graphs G_1 and G_2 introduced above where the 1-WL test failed. As shown in Figure 4 the subgraphs built around the nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$ will be structurally different, *i.e.*, triangles in $\mathcal{G}_1$ and 3 nodes path graphs in $\mathcal{G}_2$. As a consequence, the feature vectors $x(v)$ over the nodes of the two graphs will be different, allowing the GKNN to distinguish between them.

Empirical evaluation. To further strengthen the argument that our network is able to distinguish graphs where the 1-WL test fails, we empirically evaluate the ability of a GKNN to distinguish between the two graphs of Figure 4. Specifically, we train a GKNN using the WL kernel with a single mask with a maximum number of nodes set to either 3 or 4. In both cases, when we inspect the learned mask we find that its mode corresponds to the 3-nodes cycle (triangle). Indeed, the presence of triangles is what distinguishes G_2 from G_1 and allows the GKNN to discriminate between the two graphs successfully.

F. Computational complexity and runtime analysis

The computational complexity of our model is $O(NK(s))$, where N is the number of nodes of nodes of the input graph and $K(s)$ is the complexity of the kernel computation, which in turn depends on the size s of the subgraphs. Note that $s \leq N$, where equality is attained for every subgraph in the case of complete graphs. More precisely, if we limit ourselves to the case of graphs with bounded degree, we have $s = O(\min(N, d_{\max}^r))$ where $d_{\max}$ denotes the maximum node degree.

In terms of runtime, for the datasets considered in this paper, this translates to the results shown in Table I, where we report the average epoch training time and the average inference time per dataset obtained during the grid search (for the best model, for each dataset and fold). Both training and testing of each model were performed using an Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz processor and an NVIDIA Tesla V100. Note that in order to exploit the computational power of GPUs better we implemented a GPU version of the WL kernel. Although the resulting inference runtime is higher than three widely used baselines, we compared our model to (ECC [53], DiffPool [54], and GIN [32]), it is of a similar order of magnitude. When it comes to training, on the other hand, the computational bottleneck of our network is the computation of the numerical gradient, which in turn explains the subpar runtime.

G. Relation to existing works

The closest works to ours we are aware of are [13], [14]. Nikolentzos *et al*. propose a GNN where the first layer consists of a series of hidden graphs that are compared against the input graph using a random walk kernel [13]. However, due to their optimization strategy, their model only works with a single differentiable kernel, whereas our model allows us to tap into the expressive power of any type of kernel. Moreover, in the architecture of [13], the learned structural masks are assumed to be complete weighted graphs, whereas we allow for graphs with arbitrary structure. Finally, note that [14] works very much in the same way of [13], as it relies on the underlying graph kernel to be differentiable and, more specifically, it makes use of the random walk graph kernel. Another similarity with [13] is that the model proposed in [14] is also not purely structural, as opposed to ours. Instead, it learns matrices of weights which, in order to be interpreted as learned structural patterns, need to be first thresholded using a ReLU function that prunes edges associated with smaller weights.

IV. Experimental evaluation

In this section we analyze the model hyper-parameters, evaluate the proposed architecture on graph classification and regression tasks, and compare it with widely used baselines and other GNN models. A Python implementation of our model is available at https://github.com/gdl-unive/Graph-Kernel-Convolution.

A. Datasets

In our experiments we make use of 6 graph classification and 1 graph regression real-world datasets.

MUTAG [55] is a dataset consisting of 188 mutagenic aromatic and heteroaromatic nitro compounds (with 7 discrete labels) assayed for mutagenicity in Salmonella typhimurium. The goal is predicting whether each compound possesses mutagenicity effect on the Gram-negative bacterium Salmonella typhimurium.

NCI1 (the anti-cancer activity prediction dataset) is a balanced subset of datasets of chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines [56]. It has 37 discrete labels.

The PROTEINS [57], [58] dataset consists of proteins represented as graphs. Nodes represent the amino acids and an edge occurs between two vertices if they are neighbors in the amino-acid sequence or 3D space. It has 3 discrete labels, representing helix, sheet, or turn. The aim is to distinguish proteins into enzymes and non-enzymes.

The PTC (Predictive Toxicology Challenge) dataset [59], [60] is a collection of chemical compounds reporting the carcinogenicity for Male Rats (MR), Female Rats (FR), Male Mice (MM), and Female Mice (FM). We selected graphs of Male Rats (MR) for evaluation, consisting of 344 graphs very small and sparse with 19 discrete labels.

ENZYMES consists of protein tertiary structures obtained from [57], in particular 600 enzymes from the BRENDA enzyme database [61]. The task for this dataset is assigning each enzyme to one of the 6 EC top-level classes. For our experiments, we considered only 3 discrete labels.

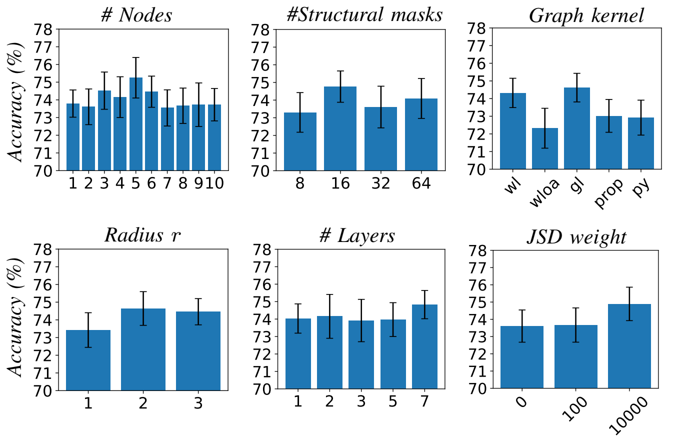


Fig. 5. Ablation study: bar plots of classification accuracy with standard error. Left to right, top to bottom: number of nodes (*i.e.*, structural mask size), number of structural masks, kernel functions (WL, WL with Optimal Assignment, Graphlet, Propagation, and Pyramid match kernel), subgraph radius, number of GKC layers, and weight of the JSD loss.

The IMDB-BINARY/MULTI datasets [62] collect graphs about movie collaboration. Each graph consists of nodes representing actors/actresses who played roles in movies in IMDB, and an edge between two vertices takes place if they appear in the same movie. Graphs are derived from the Action and Romance genres. The task is to identify which genre an ego-network graph belongs to. Note that while the bio/chemo-informatics graphs described above have categorical features, the nodes have no features in the IMDB-BINARY/MULTI datasets.

For the graph regression task, we employ a common benchmark molecular dataset, ZINC [63], a drug-constrained solubility prediction dataset. In particular, we use a subset (12K) of the ZINC molecular graphs (250K), as in [64]. Here, the goal is regressing the constrained solubility, which is the desired molecular property. The node features in each molecular graph refer to the types of heavy atoms. Even though the dataset takes into consideration the types of bonds between nodes encoded as edge features, we do not use them in our experiments.

In order to explore the ability of our model to provide an insight on the structural patterns being learned, we also built 5 further synthetic datasets based on 5 graph motifs. For more information on these datasets see Subsection IV-C.

Finally, note that in this work we restrict our attention to datasets composed of undirected graphs with categorical node attributes. However, in principle, the proposed GKNN can also operate with graph kernels that allow for directed, edge-attributed graphs. Continuous features can also be dealt with as long as the kernel is differentiable with respect to them.

B. Ablation study

We start by running an ablation study on the PROTEINS dataset, where we investigate how the various components and hyper-parameters of our architecture affect the model performance. To this end, we perform a grid search over all the

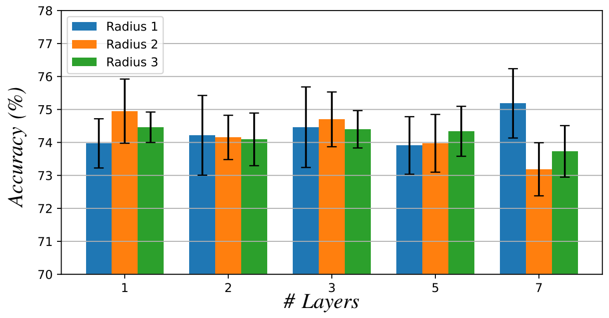


Fig. 6. Ablation study: bar plots of classification accuracy with standard error. Each group of bars shows the accuracy of models with a given depth, while the different colors within each group correspond to different subgraph radii.

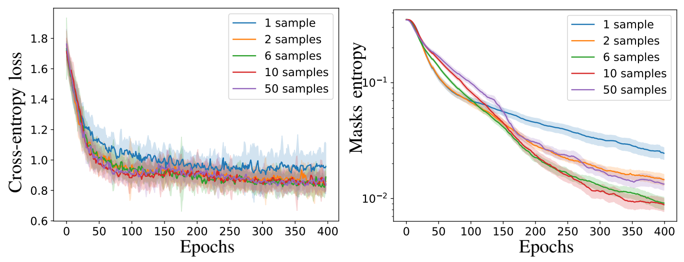


Fig. 7. Cross-entropy loss (left) and masks entropy (right) for increasing number of samples during the important sampling optimization. Each model has been trained with 8 masks for 400 epochs on the first fold of MUTAG. We report curves with standard error computed on 20 runs for each number of samples.

studied hyperparameter values where we compute the accuracy for a specific hyperparameter and value pair by averaging the results over the 3 best performing models for each fold. Specifically, we study the influence of the number of nodes (*i.e.*, the maximum size of the structural masks), number of structural masks, kernel function, subgraph radius, number of GKC layers, and weight of $loss_{JSD}$.

We report the average accuracy results as bar plots with standard error in Figure 5 (from left to right: number of nodes, number of structural masks, kernel function, subgraph radius, number of GKC layers, and weight of the JSD loss). Both the number of structural masks and the maximum number of their nodes play an important role in the classification accuracy and thus need to be carefully chosen. Note that we only fix the maximum number of nodes produced by a given mask, meaning that, in principle, at convergence, the model may have learned masks corresponding to graphs with fewer nodes. Also, while nothing in our framework prevents us from fixing a different maximum for each mask, for simplicity and to reduce the number of free parameters, we set the maximum to be the same for all masks.

As expected, the choice of the kernel function is also a crucial factor, highlighting the advantage of allowing plug-in any non-differentiable graph kernel function. Indeed, one can imagine that the choice of the kernel affects the type of structural patterns our network can capture and, therefore, impacts its overall performance. For instance, if we were to choose a kernel that does not distinguish between the presence and absence of a given structural pattern, and if this pattern was important to discriminate between two classes of

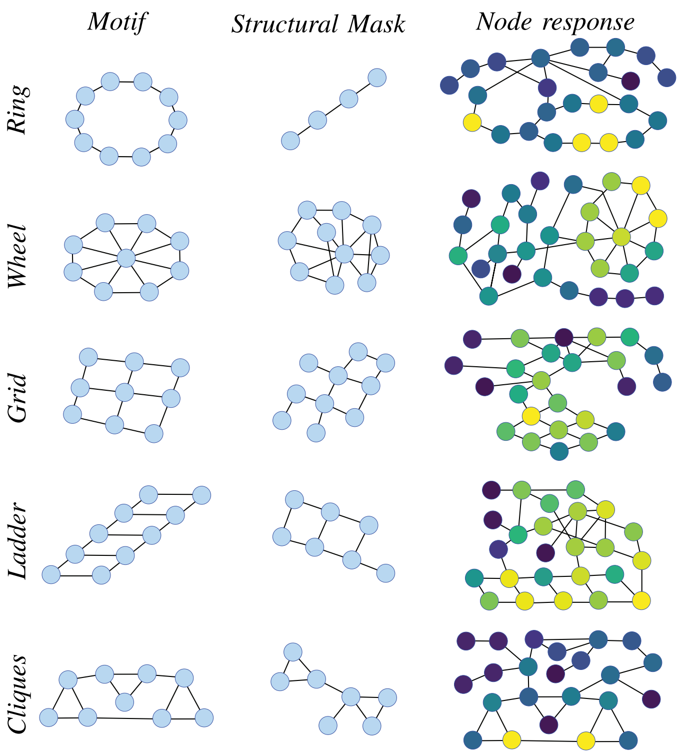


Fig. 8. Interpretability analysis results. Each row refers to a graph motif. Left: original motif. Mid: structural mask with the strongest response. Right: a sample graph including the motif. Colors indicate the node response to the filter, with lighter colors (yellow) indicating a high response and darker colors (blue) indicating a low response.

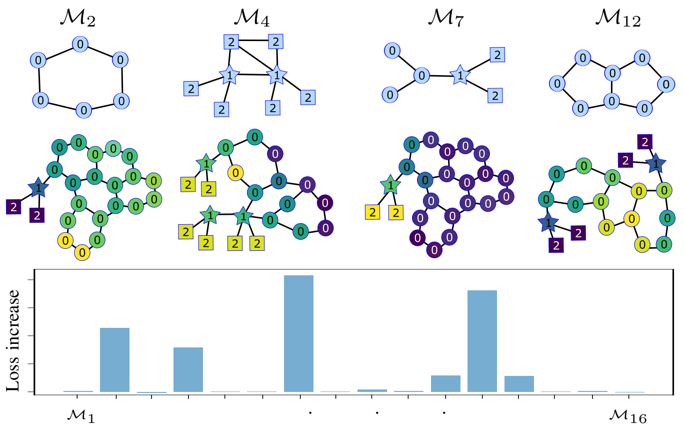


Fig. 9. Top 4 significant structural masks (top) learned by our model on the MUTAG dataset and their response over input graph nodes (middle). Shape and number both represent the node label. The bottom bar plot shows the impact of each structural mask on the classification loss.

graphs, we can foresee the performance of our network to be negatively impacted.

As for the subgraph radius, the number of neighbors clearly influences the model performance, and there seems to be an optimal subgraph size that is able to catch the structural characteristics of the input graphs. The results also confirm the importance of the JSD loss as the performance of the GKNN increases with its use.

We also study the relation between the subgraphs radius and the number of layers of the model. The results of Figure 6 show that the optimal number of layers is connected to the subgraphs radius. The models with radius 1 need more layers to perform better, while wider subgraphs tend to decrease the performance with deeper networks. Indeed, when using smaller subgraphs, more convolution layers are needed to collect information from a larger neighborhood.

Finally, Figure 7 shows the influence of the number of samples drawn during the importance sampling (see Section III-D) in terms of model performance (left) and entropy of the masks (right). The results suggest that our model is robust with respect to the choice of the sample size. Indeed, when the sample size is at least 2, its impact on the model performance appears to be negligible. At the same time, Figure 7 (right) shows that all masks converge to a low value of the entropy, regardless of the sample size. Overall, these results suggest that we can minimize the runtime by drawing a small number of samples without compromising the model performance accuracy. Accordingly, in our experiments, we fixed the sample size to 3.

C. Interpretability analysis

One of the factors that fostered interpretability in classical CNNs is the possibility to visualize and analyze the learned filters, capturing the fundamental structures characterizing the input images [13]. This feature is missing in the classical formulation of graph convolution relying on the message-passing paradigm. On the other hand, our method learns actual graph masks, potentially increasing the interpretability of the model by allowing us to probe into the explanatory factors of variation behind the input data through the learned structural masks.

To show the potential of our model to capture fundamental structures in the input graphs, we devised a simple but effective synthetic experiment. First, we train our model on a binary classification task, *i.e.*, predicting if a certain graph motif is present or not in the given graph. Then, we qualitatively assess whether the learned structural masks have captured the graph motif. Following the setup of [13], we created 5 different datasets, one for each of the graph motifs depicted in Figure 8 (leftmost column). Each dataset is composed of 2,000 graphs, equally divided into positive and negative samples, with a varying number of nodes between 30 and 50. Each sample pair is generated starting from the same Erdős–Rényi graph, with an edge observation probability of 0.1. Next, two new graphs are built for each Erdős–Rényi graph. The first one, labeled as 1, is created by connecting each node of the Erdős–Rényi to the nodes of the structural motif with probability 0.02. The second one, labeled as 0, is obtained in a similar manner; however, instead of a motif, we insert a random graph with the same number of nodes and edges of the motif structure. With these datasets to hand, the model is then trained end-to-end to predict the graph labels on a 90/10$\%$ train/test split. Note that we trained the network with the same hyper-parameters as in the ablation baseline in section IV-B, with the exception of the number of structural masks, which is set to 8.

The results show that the learned structural masks (*i.e.*, the mode of the learned edge and label probabilities) have very similar structures to those of the corresponding motifs, with only a few missing or misplaced edges (see for instance the columns corresponding to the ring and the wheel). Moreover, the distribution of the responses in the original graphs clearly highlights the motifs position. Overall, the results demonstrate that our model is able to extract the salient structural patterns, which in turn allows us to understand what features were detected for a given input graph. We would like to stress that even though this aspect is not the final goal of this work, the interpretability of our model definitely helps to foster trust in our approach.

To further investigate the ability of our model to capture salient structural features, we show in Figure 9 some of the structural masks learned on the MUTAG dataset. In particular, we trained a model with the hyper-parameters of the baseline architecture used in the ablation study of Section IV-B. To select the most significant filters, we manually set to zero the response of the ith filter $\mathcal{M}_i$ and evaluate again the classification loss. In the bar plot (Figure 9, bottom), we report the loss increase after zeroing each filter response. Below each mask, we also show the input graph of the training set that gave the highest kernel response among all nodes, together with the per-node response as node colors (high response in yellow, low response in blue). In summary, our analysis suggests that the learned masks can, to some extent, be interpreted as discriminative structural patterns that allow our network to distinguish between different classes of graphs.

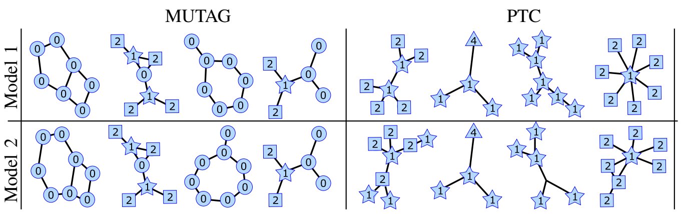


Fig. 10. Top 4 significant structural masks learned by our model on the MUTAG (left) and PTC (right) datasets starting from two different random initializations of the model weights (rows). Shape and number both represent the node label.

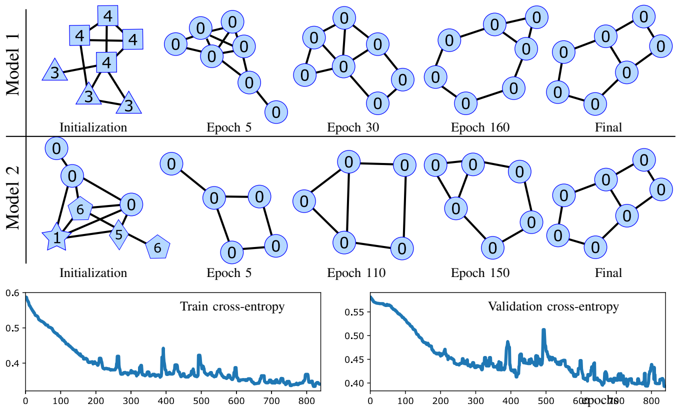


Fig. 11. Top: Evolution of the learned structural masks during the training epochs showing two different initializations (belonging to two different models) that led to the same structural mask on the MUTAG dataset. Bottom: Cross-entropy loss during training.

TABLE II
CLASSIFICATION RESULTS ON CHEMO/BIO-INFORMATICS AND SOCIAL DATASETS. MEAN ACCURACY (ACC) OR MEAN ABSOLUTE ERROR (MAE) AND STANDARD ERROR ARE REPORTED. THE BEST PERFORMANCE (PER DATASET) IS HIGHLIGHTED IN BOLD, AND THE SECOND BEST IS UNDERLINED.

	MUTAG (acc)	PTC (acc)	NCI1 (acc)	PROTEINS (acc)	IMDB-B (acc)	IMDB-M (acc)	ZINC (mae)
Baseline	78.57 ± 4.00	58.34 ± 2.02	68.50 ± 0.87	73.05 ± 0.90	49.50 ± 0.79	32.00 ± 1.17	0.656 ± 0.009
WL	82.67 ± 2.22	55.39 ± 1.27	**79.32 ± 1.48**	74.16 ± 0.38	**71.80 ± 1.03**	43.33 ± 0.56	1.244 ± 0.015
DiffPool	81.35 ± 1.86	55.87 ± 2.73	75.72 ± 0.79	73.13 ± 1.49	67.80 ± 1.44	44.93 ± 1.02	0.487 ± 0.007
GIN	78.13 ± 2.88	56.72 ± 2.66	78.63 ± 0.82	70.98 ± 1.61	71.10 ± 1.65	46.93 ± 1.06	0.381 ± 0.007
DGCNN	85.06 ± 2.50	53.50 ± 2.71	76.56 ± 0.93	74.31 ± 1.03	53.00 ± 1.32	41.80 ± 1.39	0.598 ± 0.005
ECC	79.68 ± 3.78	54.43 ± 2.74	73.48 ± 0.71	73.76 ± 1.60	68.30 ± 1.56	44.33 ± 1.59	0.726 ± 0.011
GraphSAGE	77.57 ± 4.22	59.87 ± 1.91	75.89 ± 0.96	73.11 ± 1.27	68.80 ± 2.26	47.20 ± 1.18	0.462 ± 0.007
sGIN	84.09 ± 1.72	56.37 ± 2.28	77.54 ± 1.00	73.59 ± 1.47	71.30 ± 1.75	**48.60 ± 1.48**	-
KerGNN	82.43 ± 2.73	53.15 ± 1.83	74.16 ± 3.36	72.96 ± 1.46	67.90 ± 1.33	46.87 ± 1.43	0.568 ± 0.007
RWGNN	82.51 ± 2.47	55.47 ± 2.70	72.94 ± 1.16	73.95 ± 1.32	69.90 ± 1.32	46.20 ± 1.08	0.547 ± 0.004
Ours (WL)	**85.73 ± 2.70**	59.29 ± 2.54	73.63 ± 1.32	74.94 ± 1.10	69.70 ± 2.20	47.87 ± 1.78	**0.364 ± 0.005**
Ours (GL)	85.24 ± 2.28	**60.13 ± 1.94**	71.51 ± 1.20	**75.36 ± 1.12**	69.90 ± 1.44	45.67 ± 1.22	-

D. Convergence analysis

To experimentally assess the convergence of our optimization strategy, we investigated the behavior of the structural masks both at convergence and during the optimization. In Figure 10, we show the most significant (see IV-C for the definition) structural masks learned by two different training instances of the same architecture. Both models end up learning similar structural masks, indicating that our optimization is able to reach a stable local optimum. This is also supported by the interpretability analysis of Section IV-C, showing that the learned masks resemble the local sub-structures of the input graphs.

In Figure 11, we also investigate the evolution of different initializations of the structural mask weights leading to the same final structure in two different models trained on MUTAG. Node labels have a faster convergence than connectivity, which is probably due to their bigger impact on the similarity score computed by graph kernels (WL in this specific case). The bottom of the figure also reports the decreasing curves of the cross-entropy losses during training on both train and validation sets.

E. Graph classification and regression

We compare our model against: • 6 state-of-the-art GNNs: ECC [53], DGCNN [46], DiffPool [54], GIN [32], (s)GIN [65], and GraphSAGE [66]; • two distinct baselines, depending on the dataset type (see below): Molecular Fingerprint [67], [68] and Deep Multisets [69]; • the WL kernel [7]; • RWNN [13] and KerGNN [14], two GNN models employing differentiable graph kernels, the closest existing neural architectures to ours. We use a C-SVM [70] classifier for the WL subtree kernel. For the baselines, we follow [71] and use the Molecular Fingerprint technique [67], [68] for chemical datasets. For the social dataset, we rely on the permutation invariant model of [69].

To ensure a fair comparison, we follow the same experimental protocol for each of these methods. We perform 10-fold cross validation where in each fold the training set is further subdivided in training and validation with a ratio of 9:1. The validation set is used for both early stopping and to select the best model within each fold. Importantly, folds and train/validation/test splits are consistent among all the methods. Note that in our experimental setup, the best performing model on the validation set is directly evaluated on the test set without first re-training it on the union of the training and validation sets.

We perform a grid search for all the methods to optimize the hyper-parameters. In particular, for the WL method, we optimize the value of C and the number of WL iterations $h \in \{4, 5, 6, 7\}$. For the RWGNN, we investigate the hyper-parameter ranges used by the authors [13], while for all the other GNNs and the baselines, we follow [71] and we perform a full search over the hyper-parameters grid. For our model, we explore the following hyper-parameters: number of structural masks in $\{8, 16, 32\}$, maximum number of nodes of a structural mask in $\{6, 8, 10\}$, subgraph radius in $\{1, 2, 3\}$, and number of layers in $\{1, 2, 3\}$. We implemented two versions of our network, one that makes use of the WL kernel [7] and one that makes use of the Graphlet (GL) kernel [17] (the top 2 performing kernels identified in the ablation study). For the GL kernel, we use graphlets of size 5 on all datasets except for IMDB-B and IMDB-M, where we limit the size to 3[2].

In all our experiments, we train our model for 1000 epochs, using the Adam optimizer with a learning rate of 0.001 for MLP weights and 0.01 for the edit operation probabilities and a batch size of 32. The MLP takes as input the sum pooling of the node features and is composed by two layers of output dimension m and c (# classes) with a ReLU activation in-between. We use cross-entropy and mean absolute error (MAE) losses for the graph classification and regression tasks, respectively.

Results. The accuracy for each method and dataset is reported in Table II. The performance of our approach outperforms the baselines [67], [69], [68] on all the datasets, with the improvement being statistically significant on all datasets except PTC. Compared to other GNNs, the performance of our model is on par with the best ones, as we can see for MUTAG, PROTEINS, and PTC. This holds as long as the number of node labels is small. Indeed, in the case of high-dimensional node labels (*e.g.*, NCI1), the classification accuracy tends to decrease as the optimization becomes harder.

[2]This is due to the presence of high degree nodes in the IMDB datasets and the fact that the computational complexity of the GL kernel is $O(Nd_{\max}^{k-1})$, where N is the number of nodes of the graph and $d_{\max}$ is the maximum degree.

As for the social dataset, the performance of our approach is satisfactory, particularly considering the lack of discriminative sub-structures in social networks. Finally, our method achieves the best performance (*i.e.*, lowest MAE) on ZINC, with only GIN achieving a similar yet higher MAE. It should also be noted that our model always outperforms or performs comparably to both KerGNN and RWGNN, the only two other models that allow some degree of interpretability of the learned structures.

V. Conclusion

We introduced a new convolution operation on graphs based on graph kernels. We proposed the graph kernel convolution layer and an architecture that makes use of this layer for graph classification purposes. The benefits of this architecture include the definition of a fully structural model that can exploit the vast collection of existing graph kernels, a superior expressive power when compared to standard GNNs, and the possibility to visualize the learned substructures in a way that is reminiscent of the convolutional filters of standard CNNs. Future work will attempt to address the limitations of the current approach. For example, our model is also not well suited for social graphs, where structure plays a minor role, and small-worldness implies that the subgraphs will span the majority of the input graph even at small radii. It should also be easy to modify our architecture to allow any number of graph kernels to be plugged in, allowing the network to learn which kernel is best suited for the dataset and task at hand. In principle, our architecture can also operate using graph kernels for both directed and undirected graphs (*e.g.*, GL and WL kernels), as well as edge attributed ones (*e.g.*, WL kernel, categorical attributes only). In general, node and edge features are not restricted to being categorical, however in the case of continuous features the kernel should be differential with respect to the features. Note that these limitations in handling social graphs and the necessity for the kernel to be differential with respect to continuous features may limit the immediate applicability of our model in some domains. Future work will aim to address these limitations and explore our network's performance on node classification and regression tasks.

Acknowledgments

L.C. and A.B. are supported by the PRIN 2022 project n. 2022AL45R2 (EYE-FI.AI, CUP H53D2300350-0001). G.M. acknowledges financial support from the European Union - NextGenerationEU, in the framework of the iNEST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS_00000043 – CUP H43C22000540006). In this regard, the views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.

References

[1] R. P. Duin and E. Pekalska, "The dissimilarity space: Bridging structural and statistical pattern recognition," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 826–832, 2012.

[2] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: Github as a collaborative social network," in *Eighth international AAAI conference on weblogs and social media*, 2014.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations*, ser. ICLR '17, 2017.

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[5] C. Ye, C. H. Comin, T. K. D. Peron, F. N. Silva, F. A. Rodrigues, L. d. F. Costa, A. Torsello, and E. R. Hancock, "Thermodynamic characterization of networks using graph polynomials," *Physical Review E*, vol. 92, no. 3, p. 032810, 2015.

[6] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 021001, 2020.

[7] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[8] L. Bai, L. Rossi, A. Torsello, and E. R. Hancock, "A quantum jensen–shannon graph kernel for unattributed graphs," *Pattern Recognition*, vol. 48, no. 2, pp. 344–355, 2015.

[9] G. Minello, L. Rossi, and A. Torsello, "Can a quantum walk tell which is which? a study of quantum walk-based graph similarity," *Entropy*, vol. 21, no. 3, p. 328, 2019.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations*, 2018.

[11] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[12] W. Cong, M. Ramezani, and M. Mahdavi, "On provable benefits of depth in training graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[13] G. Nikolentzos and M. Vazirgiannis, "Random walk graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 211–16 222, 2020.

[14] A. Feng, C. You, S. Wang, and L. Tassiulas, "Kergnns: Interpretable graph neural networks with graph kernels," *ArXiv Preprint: https://arxiv.org/abs/2201.00491*, 2022.

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

[16] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Fifth IEEE international conference on data mining (ICDM'05)*. IEEE, 2005, pp. 8–pp.

[17] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 488–495.

[18] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 321–328.

[19] L. Bai and E. R. Hancock, "Graph kernels from the jensen-shannon divergence," *Journal of mathematical imaging and vision*, vol. 47, no. 1, pp. 60–69, 2013.

[20] L. Rossi, A. Torsello, and E. R. Hancock, "Measuring graph similarity through continuous-time quantum walks and the quantum jensen-shannon divergence," *Physical Review E*, vol. 91, no. 2, p. 022815, 2015.

[21] Y. Zhang, L. Wang, R. C. Wilson, and K. Liu, "An r-convolution graph kernel based on fast discrete-time quantum walk," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 1, pp. 292–303, 2020.

[22] G. Da San Martino, N. Navarin, and A. Sperduti, "Tree-based kernel for graphs with continuous attributes," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3270–3276, 2017.

[23] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, pp. 1–42, 2020.

[24] 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, 2008.

[25] 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.

[26] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[27] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[28] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 1993–2001.

[29] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[30] L. Bai, L. Cui, Y. Jiao, L. Rossi, and E. Hancock, "Learning backtrackless aligned-spatial graph convolutional networks for graph classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[31] Z. Li, H. Liu, Z. Zhang, T. Liu, and N. N. Xiong, "Learning knowledge graph embedding with heterogeneous relation attention networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3961–3973, 2021.

[32] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[33] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 4602–4609.

[34] A. Bicciato, L. Cosmo, G. Minello, L. Rossi, and A. Torsello, "Classifying me softly: A novel graph neural network based on features soft-alignment," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2022, pp. 43–53.

[35] ——, "Gnn-lofi: A novel graph neural network through localized feature-based histogram intersection," *Pattern Recognition*, vol. 148, p. 110210, 2024.

[36] M. Eliasof, E. Haber, and E. Treister, "Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[37] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron, "Equivariant subgraph aggregation networks," in *International Conference on Learning Representations*, 2022.

[38] G. Bouritsas, F. Frasca, S. P. Zafeiriou, and M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[39] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[40] T. Lei, W. Jin, R. Barzilay, and T. Jaakkola, "Deriving neural architectures from sequence and graph kernels," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2024–2033.

[41] G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis, "Kernel graph convolutional neural networks," in *International conference on artificial neural networks*. Springer, 2018, pp. 22–32.

[42] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5723–5733, 2019.

[43] D. Chen, L. Jacob, and J. Mairal, "Convolutional kernel networks for graph-structured data," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1576–1586.

[44] G. Bouritsas, A. Loukas, N. Karalias, and M. M. Bronstein, "Partition and code: learning how to compress graphs," *arXiv:2107.01952*, 2021.

[45] N. Navarin, D. Van Tran, and A. Sperduti, "Learning kernel-based embeddings in graph neural networks," in *ECAI 2020*. IOS Press, 2020, pp. 1387–1394.

[46] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[47] D. Nebel, M. Kaden, A. Villmann, and T. Villmann, "Types of (dis-) similarities and adaptive mixtures thereof for improved classification learning," *Neurocomputing*, vol. 268, pp. 42–54, 2017.

[48] E. Englesson and H. Azizpour, "Generalized jensen-shannon divergence loss for learning with noisy labels," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30 284–30 297, 2021.

[49] L. Bai, L. Rossi, L. Cui, J. Cheng, and E. R. Hancock, "A quantum-inspired similarity measure for the analysis of complete weighted graphs," *IEEE transactions on cybernetics*, vol. 50, no. 3, pp. 1264–1277, 2019.

[50] M. V. Ramana, E. R. Scheinerman, and D. Ullman, "Fractional isomorphism of graphs," *Discrete Mathematics*, vol. 132, no. 1-3, pp. 247–265, 1994.

[51] R. Sato, "A survey on the expressive power of graph neural networks," *arXiv preprint arXiv:2003.04078*, 2020.

[52] G. Nikolentzos, G. Dasoulas, and M. Vazirgiannis, "k-hop graph neural networks," *Neural Networks*, vol. 130, pp. 195–205, 2020.

[53] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[54] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[55] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[56] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 347–375, 2008.

[57] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[58] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.

[59] C. Helma, R. D. King, S. Kramer, and A. Srinivasan, "The predictive toxicology challenge 2000–2001," *Bioinformatics*, vol. 17, no. 1, pp. 107–108, 2001.

[60] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," *arXiv preprint arXiv:1206.6483*, 2012.

[61] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "Brenda, the enzyme database: updates and major new developments," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D431–D433, 2004.

[62] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[63] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman, "Zinc: a free tool to discover chemistry for biology," *Journal of chemical information and modeling*, vol. 52, no. 7, pp. 1757–1768, 2012.

[64] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[65] X. Di, P. Yu, R. Bu, and M. Sun, "Mutual information maximization in graph neural networks," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[66] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[67] L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi, "Graph kernels for chemical informatics," *Neural networks*, vol. 18, no. 8, pp. 1093–1110, 2005.

[68] E. Luzhnica, B. Day, and P. Liò, "On graph classification networks, datasets and baselines," *arXiv preprint arXiv:1905.04682*, 2019.

[69] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 3394–3404.

[70] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

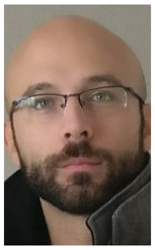
[71] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Luca Cosmo is an Assistant Professor in Computer Science at University Ca' Foscari of Venice, working in Geometric Deep Learning and Generative AI, with more than 50 published papers. He has been serving on the program committee and as a reviewer for top conferences and journals in these fields and he gave short courses and tutorials in geometry processing related topics at major conferences in the field. He is a member of ELLIS and co-founded a start-up working on geometry processing for industrial product inspection.

Giorgia Minello Giorgia Minello worked in the industry for almost ten years before receiving, in 2019, her PhD in Computer Science from Ca' Foscari, University of Venice (Italy), where she currently holds a Postdoctoral Research Fellow position. She is also a scientific collaborator for the IESE Business School, University of Navarra, in the context of a social network analysis project. Her research mainly focuses on Structural Pattern Recognition, Machine Learning and Natural Language Processing. The leading topic of her studies concerns the analysis of networks, such as the use of quantum approaches for structural analysis or the application of network analysis methods for mining textual corpora.

Alessandro Bicciato received the PhD in computer science from the Ca' Foscari University of Venice, Italy in 2024. He is currently working as a post-doctorate researcher at the Computer Science Department of Ca' Foscari University of Venice, Italy. His research interests include network science and pattern recognition.

Michael Bronstein Michael Bronstein is the DeepMind Professor of AI at the University of Oxford. He previously served as Head of Graph Learning Research at Twitter, professor at Imperial College London, and held visiting appointments at Stanford, MIT, and Harvard. He is the recipient of the Royal Society Wolfson Research Merit Award, Royal Academy of Engineering Silver Medal, Turing World-Leading AI Research Fellowship, five ERC grants, two Google Faculty Research Awards, and two Amazon AWS ML Research Awards. He is a Member of the Academia Europaea, Fellow of IEEE, IAPR, BCS, and ELLIS, ACM Distinguished Speaker, and World Economic Forum Young Scientist. In addition to his academic career, Michael is a serial entrepreneur and founder of multiple startup companies, including Novafora, Invision (acquired by Intel in 2012), Videocites, and Fabula AI (acquired by Twitter in 2019). He is the Chief Scientist at VantAI and scientific advisor at Recursion Pharmaceuticals.



Emanuele Rodolà is Professor of Computer Science at Sapienza University of Rome, where he leads the GLADIA group of Geometry, Learning and Applied AI, funded by an ERC Grant and a Google Research Award. He is currently an ELLIS fellow, and formerly fellow of the Humboldt Foundation and the Japan Society for the Promotion of Science. His research interests lie at the intersection of geometry processing, graph theory, geometric deep learning, applied AI and computer vision, and has published more than 100 papers in these areas.

Luca Rossi received the PhD degree in computer science from the Ca' Foscari University of Venice, Italy, in 2013. He is currently an Assistant Professor with the Hong Kong Polytechnic University, having held various positions with the University of Birmingham, Aston University, Southern University of Science and Technology, and Queen Mary University of London. He has published more than 50 papers in international journals and conferences. His research interests include the areas of pattern recognition, data mining, and network science. He is currently a member of the editorial board of the journal Pattern Recognition and vice-chair of the technical committee 2 of the International Association for Pattern Recognition.

Andrea Torsello received his PhD in computer science at the University of York, UK. From 2007 he is with Ca'Foscari University of Venice, Italy, where he is Full Professor. He has published over 150 papers in international journals and conferences. His research interests are in the areas of Computer Vision and Pattern Recognition, in particular the interplay between Stochastic and Structural approaches as well as Game-Theoretic and Physical models. He is currently a member of the editorial board of the journal Pattern Recognition and chair of the technical committee 2 of the International Association for Pattern Recognition.