# GRAPH KERNEL NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## 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 and number 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 impact of the model hyper-parameters and we show that our model achieves competitive performance on standard graph classification datasets.

## 1 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, from collaborations (Lima et al., 2014; Kipf & Welling, 2017) to biological (Gilmer et al., 2017; Ye et al., 2015) and physical (Shlomi et al., 2020) 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 (Shervashidze et al., 2011; Bai et al., 2015; Minello et al., 2019). 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 (Kipf & Welling, 2017; Gilmer et al., 2017; Veličković et al., 2018)

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 a vector 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, as a result of 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 generalised 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. In this paper we propose a neural architecture that bridges the two worlds of graph kernels and GNNs. The main idea is 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 generalise 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 sub-graphs. These in turn can offer better intrepretability, in the form of insights into what structural patterns in the input graphs are related to the corresponding output.

**Our main contributions are:** 1. Unlike existing approaches that require embedding the input graph into a larger, relaxed space with a higher likelihood of ending up in a local minimum, our model is fully structural; 2. Our architecture allows to plug-in any number and type of graph kernel (not just a single differentiable kernel as in Nikolentzos & Vazirgiannis (2020)); 3. As an added benefit, our model provides some interpretability in terms of the structural masks that are learned during the training process, similarly to convolutional masks in traditional CNNs; 4. Finally, we analyse the expressive power of our model and we show that it is greater than that of standard message-passing GNNs and the equivalent Weisfeiler–Lehman (WL) graph isomorphism test.

The remainder of this paper is organised as follows. In Section 2, we review the related work. In Section 3, we present our neural architecture where graph kernels are used to redefine convolution on graph data. In Section 4, we investigate the hyper-parameters of our architecture through an extensive ablation study, provide some insight on the interpretability of the structural masks, and evaluate our architecture on standard graph classification benchmarks. Finally, Section 5 concludes the paper.

## 2 Related work

The majority of graph kernels belongs 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 sub-structures and then counting the number of isomorphic sub-structures between the two input graphs. Depending on the type of sub-structure considered, one can build a multitude of different kernels, e.g., sub-trees (Ramon & Gärtner, 2003), shortest paths (Borgwardt & Kriegel, 2005), and graphlets (Shervashidze et al., 2009). Information propagation kernels, on the other hand, includes methods where pairs of input graphs are compared based on how information diffuses on them. Examples include random walk kernels (Kashima et al., 2003; Bai & Hancock, 2013), quantum walk kernels (Bai et al., 2015; Rossi et al., 2015), and kernels based on iterative label refinements (Shervashidze et al., 2011). While some kernels work only on undirected and unattributed graphs, other kernels are designed to handle attributes as well, either discrete- or continuous-valued (Shervashidze et al., 2011; Da San Martino et al., 2017). For a detailed review and historical perspective on graph kernels, we refer the reader to the recent survey of Kriege et al. (2020).

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. (2008), where an information diffusion mechanism is used to learn the nodes' latent representations by exchanging neighbourhood information. Depending on the form this diffusion takes, Bronstein et al. (2021) distinguish between convolutional (Kipf & Welling, 2017; Atwood & Towsley, 2016; Levie et al., 2018), attentional Veličković et al. (2018), or message passing (Gilmer et al., 2017) types of GNNs, with the latter being the most general and formally equivalent to the Weisfeiler-Lehman graph isomorphism test under some technical conditions (Xu et al., 2018; Morris et al., 2019). For a comprehensive survey of GNNs we refer the reader to Wu et al. (2020).

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. (2017), introduce a class of deep recurrent neural architectures show that this lies in the reproducing kernel Hilbert space (RKHS) of graph kernels. Nikolentzos et al. (2018) compute continuous embeddings of graphs using kernels and plug them into a neural network. Xu et al. (2018); Morris et al.

(2019) show that GNNs have the same expressiveness as the WL graph kernel (Shervashidze et al., 2011) and propose a new generalised architecture with increased expressive power. Du et al. (2019) take a somewhat different approach and exploit neural networks to introduce a new graph kernel 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. (2020) propose a graph neural architecture where each layer enumerates local sub-structures around each node and then maps them to a RKHS via a Gaussian kernel mapping. In the context of graph compression, Bouritsas et al. (2021) learn how to best decompose the graph into small substructures that are also learned.

The closest work to ours we are aware of is that of Nikolentzos & Vazirgiannis (2020), who 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. However, due to their optimisation strategy, their model only works with a single differentiable kernel, whereas our model allows us to tap into the expressive power of any type and number of kernels. Moreover, in the architecture of Nikolentzos & Vazirgiannis (2020) the learned structural masks are assumes to be complete weighted graphs, whereas we allow for graphs with arbitrary structure.

## 3 Graph Neural Networks from Graph Kernels

### 3.1 Preliminaries

**Kernel methods** are a class of algorithms that are capable of learning when presented with a particular pairwise similarity 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 2), which are nothing but positive semi-definite pairwise similarity measures on graphs. These can be either implicit (only $\mathcal{K}$ is computed) or explicit ($\phi$ is also computed).

**Weisfeiler-Lehman test.** The WL kernel (Shervashidze et al., 2011) 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 that of partitioning 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 $\mathbf{D}$. 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 try to find an embedding $Z = \square(g_\theta(X, E))$ by performing message passing (Gilmer et al., 2017) in the 1-hop neighborhood $\mathcal{N}^1_{\mathcal{G}}(v)$ of each node $v \in V$, i.e.,

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

The convolution is usually repeated for $L$ layers, and the final vectorial representation $Z$ is obtained by applying a node-wise permutation invariant aggregation operator $\square$ on the node features $z(v)$.

### 3.2 Proposed architecture

The core feature of the proposed model is the definition of a new approach to perform the convolution operation on graphs. As opposed to 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.

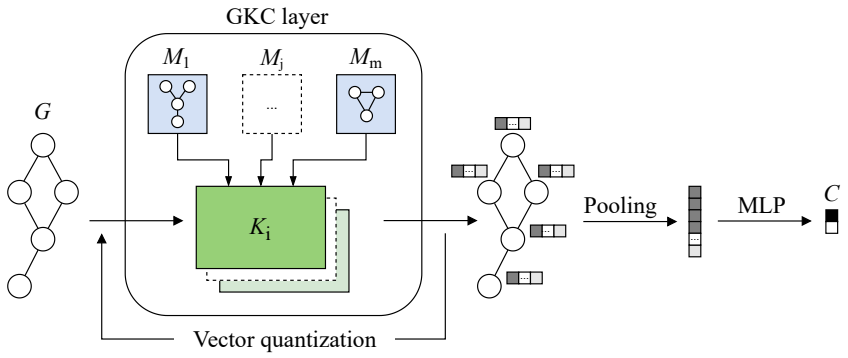


Figure 1: The proposed GKNN architecture. The input graph is fed into one or more GKC layers, where sub-graphs centered at each node are compared to a series of structural masks through a kernel function. The output is a new set of real-valued feature vectors associated to the graph nodes, which goes through a vector quantization operation. 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.

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

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

where the resulting feature vector $x(v)$ is a non-negative real valued $m$-dimensional vector collecting the kernel responses, and the $i$th structural mask $\mathcal{M}_i = (L_{\mathcal{M}_i}, E_{\mathcal{M}_i})$ is a graph with node set $L_{\mathcal{M}_i}$ (including node labels) and edge set $E_{\mathcal{M}_i}$.

Since in this paper we focus on graph classification, the final part of our architecture, as depicted in Figure 1, consists of applying a node-wise pooling operation on the previously computed feature vectors to obtain a global graph descriptor. This is then fed to a multilayer perceptron (MLP) layer to obtain the final classification.

### 3.3 Multi-layer architecture

Most graph kernels, including the WL kernel, work with discrete node labels. In our GKNN architecture, however, equation 2 yields an $m$-dimensional real valued vector for each graph node. Hence, we need to discretize the output feature space before being able to perform a further convolution operation. Before each convolution layer, except for the first one (unless the input node features are continuous), we add a *vector quantization* operation to the input node features, whereby the feature space is discretized using $k$-means clustering on the node features of the current batch. Nodes are then labeled according to the cluster index they belong to. To keep the labeling consistent among different batches, the $k$-means algorithm centroids at step $\ell + 1$ are initialized using the centroids computed at the previous step $\ell$, i.e.,

$$x_i^{\ell+1}(v) = \mathcal{K}(\mathcal{M}_i, \mathcal{N}^r_{\mathcal{G}(V, vq(X^\ell), E)}(v)), \tag{3}$$

where $vq(X)$ indicates the vector quantization operation described above.

Note that in our current implementation the gradient does not flow through the input labels. To overcome this limitation and allow the optimization of the structural masks in each layer, we add skip connections between the layers. The final graph embedding after $L$ layers is thus obtained by $\square(Z^1|\ldots|Z^L)$, where $Z^\ell$ are the output features of the $\ell$th layer and $\cdot|\cdot$ indicates the concatenation of node-wise features.

### 3.4 Optimization strategy

Our learning problem can be formulated as follows: given a set $\{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$ of $B$ training input graphs with node labels belonging to the dictionary $\mathbf{D}$ and associated class labels $y_1 \ldots y_B$, our

goal is to find the optimal parameters ($\theta$ and masks) of the model $h$ for the following minimization problem,

$$\min_{\mathcal{M}_1 \dots \mathcal{M}_m, \theta} \sum_{i=1}^{B} \textit{CrossEntropy}(h_{\mathcal{M}_1 \dots \mathcal{M}_m, \theta}(\mathcal{G}_i), y_i). \tag{4}$$

There are two main challenges that we need to consider when optimizing the structural masks. First, the number of nodes of the sub-graphs is not fixed and can in principle vary from 1 (for isolated nodes) to $n$ (for fully connected graphs). 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 kernels representation** When defining the space of graphs on which we want to optimize the structural masks $\mathcal{M}$, we have to consider the possible sub-structures 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 $d$ for each substructure and optimize for structural masks in the space

$$\mathcal{M} \in \bigcup_{p=1}^{d} \left( \tilde{\mathbf{G}}_p \times \mathbf{D}^p \right) , \tag{5}$$

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 4.

**Structural kernels 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 use a Discrete Randomized Descent (DRD) strategy. Given an initial structural mask $\mathcal{M}$, we let the graph evolve to $\mathcal{M}'$ through edit operations minimizing a cost function. These operations consist of adding or removing some edges, and changing node labels. After the editing operation we extract the maximum connected component in $\mathcal{M}'$, thus allowing us to consider all the graphs with nodes $p \leq d$ without explicitly optimizing over $p$.

The DRD update is performed during the backpropagation phase of the model training step. Through the chain rule mechanism we can estimate the discrete gradient of the final classification loss w.r.t. the structural mask $\mathcal{M}$ as $\frac{\delta loss}{\delta \mathcal{M}} = \sum_v \frac{\delta loss}{\delta x(v)} \cdot \frac{\delta x(v)}{\delta \mathcal{M}}$. The discrete sub-gradient of the kernel response $x(v)$ w.r.t. $\mathcal{M}$ can be estimated as the difference between the kernel responses after and before an edit operation:

$$\frac{\delta loss}{\delta \mathcal{M}} = \sum_v \frac{\delta loss}{\delta x(v)} \cdot \left( \mathcal{K}(\mathcal{M}', \mathcal{N}_G^r(v)) - \mathcal{K}(\mathcal{M}, \mathcal{N}_G^r(v)) \right). \tag{6}$$

During each backpropagation step, we thus sample an edit operation $e$ from the set of all edit operations on $E_{\mathcal{M}}$, and accept the edit only if the value of equation 6 is lower than or equal to 0. This means that we always evolve toward a graph locally minimizing the loss function.

Together with the structural mask $\mathcal{M}$, we also optimize a probability distribution over the edit operations. The derivative of an edit operation probability $p(e)$ is estimated using the same equation 6 and optimized with standard gradient descent optimizers. Further details are in the Appendix A.1.

**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 avoid 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 $P_i = \{\alpha x(v)_i | v \in V\}$ be the probability distribution induced by the $i$th kernel over the graph nodes. $\alpha$ is a scaling factor ensuring that $\sum_v P_i(v) = 1$. We define the JSD loss as

$$loss_{JSD} = -H\left(\sum_{i=1}^{n} P_i\right) + \sum_{i=1}^{n} H\left(P_i\right) , \tag{7}$$

where $H(P)$ is the Shannon entropy. The final loss we optimize is then the sum of 1) the CrossEntropy loss defined in equation 3.4 and 2) $loss_{JSD}$ multiplied by a weighting factor.

### 3.5 Expressive power

In this subsection, we study the expressive power of the proposed architecture. Recall from Morris et al. (2019) that the expressive power of standard GNNs (i.e., GNNs that only consider the immediate neighbours of a node when updating the labels) is equivalent to that of the 1-WL test. We argue that GKNN has a higher expressive power than the 1-WL test (and thus standard message-passing GNNs). To show this, we start by demonstrating that every pair of graphs that can be distinguished by the 1-WL test can also be distinguished by our model. Without loss of generality, in the following we consider two unlabelled graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_1 = (V_2, E_2)$ with the same number of nodes $n = |V_1| = |V_2|$.

**Theorem 1.** *Given two input graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, if the 1-WL test can distinguish between them then there exists an instance of a GKNN that can also distinguish them.*

A sketch of proof of Theorem 1 can be found in Appendix A.2. Next, we show that the GKNN can distinguish between pairs of graphs where the 1-WL test fails. To see that this is the case, consider the standard example of two graphs with 6 nodes, where the first graph is made of two disconnected cycles with 3 nodes, while the second graph is a single cycle with 6 nodes. Both graphs are regular, with every node of the graphs being adjacent to two other nodes, i.e., every node has degree 2. As explained in Nikolentzos et al. (2020), 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. As the following lemma shows, this is not an issue for the GKNN:

**Lemma 2.** *The GKNN is able to identify graph containing triangles.*

A sketch of proof of Lemma 2 can be found in Appendix A.2. From Theorem 1 and Lemma 2 it follows that the GKNN has a higher expressive power than the 1-WL test and thus standard GNNs.

## 4 Experimental evaluation

In this section we analyze the model hyper-parameters, we evaluate the proposed architecture on the graph classification task and compare it with widely used baselines and GNN models.

### 4.1 Ablation study

We performed an ablation study to investigate how the various components and hyper-parameters of our architecture affect the model performance. To this end, we consider the MUTAG dataset (for more details on the datasets, see Appendix A.4). We start by setting a baseline network configuration (one layer, 16 structural masks of 6 nodes each, subgraphs radius of 3, WL graph kernel and JSD weight of $10^{-4}$) and then we let the considered hyper-parameter(s) vary. 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 accuracy results as bar plots with standard error in Figure 2 (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 a crucial role in the classification accuracy and thus need to be carefully chosen. As expected, also the choice of the kernel function is a crucial factor, highlighting the advantage of allowing to plug-in any non-differentiable graph kernel function. As for the subgraph radius, the

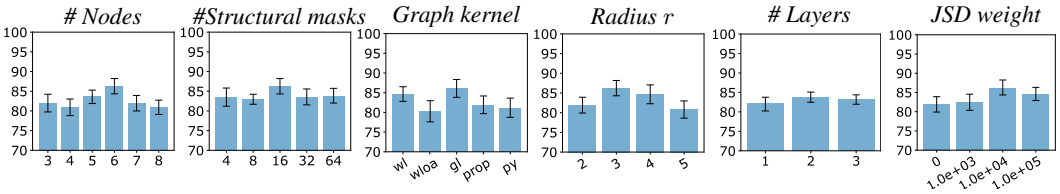


Figure 2: Ablation study: bar plots of classification accuracy with standard error. Left to right: number of nodes (i.e., structural mask size), number of structural masks, kernel functions (WL, WL with Optimal Assignment, Graphlet, Propagation, and Pyramid match kernel), subgraph radius, number of GKC layers, and weight of the JSD loss.

number of neighbors clearly influences the model performance and there seem to be an optimal subgraph size able to catch the structural characteristics of the input graphs. The ablation validates also our multilayer architechture design, even if the model requires just two layers to reach the best accuracy. This is justified by the wider neighborhood involved in our convolution operation compared to the standard message passing formulation. The results also confirm the importance of the JSD loss, as the performance of the GKNN increases once the loss is introduced.

### 4.2 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 (Nikolentzos & Vazirgiannis, 2020). 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 if the learned structural masks have captured or not the graph motif. Following the setup of Nikolentzos & Vazirgiannis (2020), we created 5 different datasets, one for each of the graph motif depicted in Figure 3 (upper row). A detailed description of the synthetic datasets and training details can be found in the Appendix A.3. The model is then trained end-to-end to predict the graph labels on a 90/10$\%$ train/test split.

The results show that the learned structural masks 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 of the input graphs, we show in Figure 4 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 4.1. To select the most significant filters, we manually set to zero the response of the $i$th filter $\mathcal{M}_i$ and evaluate again the classification loss. In the bar plot (Figure 4, left) 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 higher kernel response among all nodes together with the per-node response as node colors (yellow indicating a high response, dark blue indicating a low response).

### 4.3 Graph classification results

**Datasets** We evaluate the performance of the proposed model on the graph classification task. We make use of publicly available datasets frequently employed in the GNNs literature (Kersting et al., 2016). In particular, we use 4 bio/chemo-informatics datasets collecting molecular graphs (MUTAG,

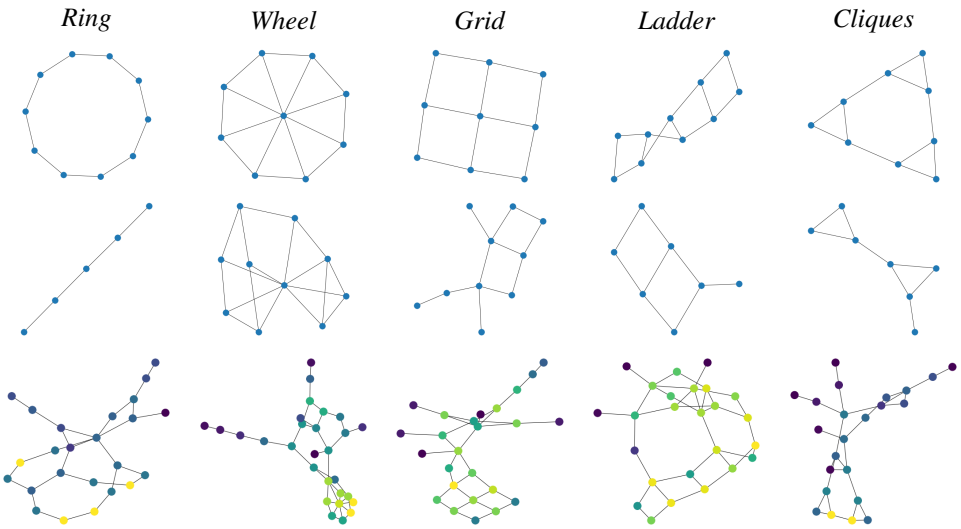


Figure 3: Interpretability analysis results. Each column refers to a graph motif. Upper row: original motif. Mid row: structural mask with the strongest response. Bottom row: a sample graph including the motif. Colors indicate the node response to the filter, with lighter colours (yellow) indicating a high response and darker colors (blue) indicating a low response.

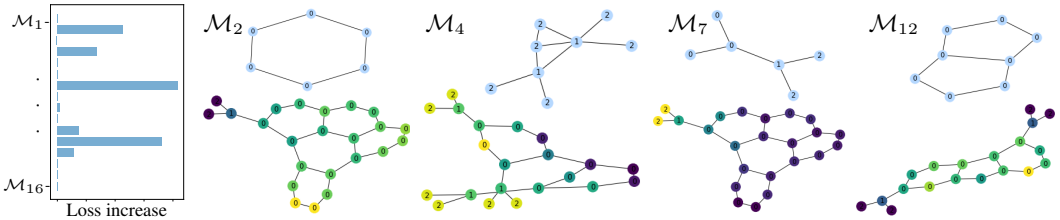


Figure 4: Top 4 significant structural masks (top) learned by our model on the MUTAG dataset and their response over input graph nodes (bottom). The left bar plot show the impact of each structural mask to the classification loss.

NCI1, PROTEINS, and PTC) and a social dataset (IMDB-BINARY). Bio/chemo-informatics graphs differ from social graphs as in the former nodes have categorical input features, whereas in the latter there are no features. More details on these datasets can be found in Appendix A.4.

**Experimental setup** We compare our model against: • 5 state-of-the-art GNNs: DGCNN (Zhang et al., 2018), DiffPool (Ying et al., 2018), GIN (Xu et al., 2018), and (s)GIN (Di et al., 2020); • two distinct baselines, depending on the dataset type (see below): Molecular Fingerprint (Ralaivola et al., 2005; Luzhnica et al., 2019) and Deep Multisets (Zaheer et al., 2017); • the WL kernel (Shervashidze et al., 2011); • RWNN, a GNN model employing a differentiable graph kernel (Nikolentzos & Vazirgiannis, 2020) and thus the closest existing neural architecture to ours. In particular, for the WL subtree kernel we use a $C$-SVM (Chang & Lin, 2011) classifier. For the baselines, as suggested in Errica et al. (2020), for chemical datasets we use the Molecular Fingerprint technique, following Ralaivola et al. (2005) and Luzhnica et al. (2019). For the social dataset we rely instead on the permutation invariant model of Zaheer et al. (2017).

With a view to achieving a fair comparison, for each of these methods we follow the same experimental protocol. In particular, 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. For all the methods we perform grid search 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 (Nikolentzos & Vazirgiannis, 2020), while for all the

Table 1: Classification results of the proposed model and other methods on the chemo/bio-informatics and social datasets. Mean accuracy and standard error are reported. Best performance (per dataset) is highlighted in bold.

| | MUTAG | PTC | NCI1 | PROTEINS | IMDB |
|---|---|---|---|---|---|
| Baseline | 78.57 ± 4.00 | 58.34 ± 2.02 | 68.50 ± 0.87 | 73.05 ± 0.90 | 49.50 ± 0.79 |
| WL | 82.67 ± 2.22 | 55.39 ± 1.27 | **79.32 ± 1.48** | 74.16 ± 0.38 | **71.80 ± 1.03** |
| DiffPool | 81.35 ± 1.86 | 55.87 ± 2.73 | 75.72 ± 0.79 | 73.13 ± 1.49 | 67.80 ± 1.44 |
| GIN | 78.13 ± 2.88 | 56.72 ± 2.66 | 78.63 ± 0.82 | 70.98 ± 1.61 | 71.10 ± 1.65 |
| DGCNN | 85.06 ± 2.50 | 53.50 ± 2.71 | 76.56 ± 0.93 | 74.31 ± 1.03 | 53.00 ± 1.32 |
| GraphSAGE | 77.57 ± 4.22 | **59.87 ± 1.91** | 75.89 ± 0.96 | 73.11 ± 1.27 | 68.80 ± 2.26 |
| sGIN | 84.09 ± 1.72 | 56.37 ± 2.28 | 77.54 ± 1.00 | 73.59 ± 1.47 | 71.30 ± 1.75 |
| RWGNN | 82.51 ± 2.47 | 55.47 ± 2.70 | 72.94 ± 1.16 | 73.95 ± 1.32 | 69.90 ± 1.32 |
| **Ours** | **85.73 ± 2.70** | 58.39 ± 3.40 | 71.52 ± 1.12 | **74.48 ± 1.10** | 69.70 ± 2.20 |

other GNNs we follow Errica et al. (2020) 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 $\{6,8\}$, subgraph radius in $\{1,2,3\}$, number of layers in $\{1,2,3\}$ and always use WL as graph kernel. In all our experiments, we train the model for $1000$ epochs, using the Adam optimizer with learning rate of $0.001$ for MLP weights and of $0.01$ 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.

**Results** The accuracy for each method and dataset is reported in Table 1. For the bio/chemo-informatics datasets, the performance of our approach 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 low. Indeed, in the case of high-dimensional node labels (e.g., NCI1), the classification accuracy tends to decrease as the optimization becomes harder. As for the social dataset, the performance of our approach is satisfactory, particularly considering the lack of discriminative substructures in social networks.

## 5 Conclusion

In this paper 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 provably 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, the optimization strategy we employed in this work prefers low-dimensional discrete input labels over high-dimensional continuous ones. Our model is also not well suited for social graphs, where structure plays a minor role and small-worldness implies that the sub-graphs will span the majority of the input graph even at small radii. Finally, the lack of widely available GPU implementations of graph kernels implies that our model cannot presently tap into the computational power of these units.

## Reproducibility Statement

To maximize reproducibility of our results by the academic community, we provide an extensive discussion of the model, its hyper-parameters, the training and the experimental setup. We also make use of widely used publicly available datasets. Finally, we include the source code in the supplementary material and we will make it publicly available upon acceptance.

## REFERENCES

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pp. 1993–2001, 2016.

Lu Bai and Edwin R Hancock. Graph kernels from the jensen-shannon divergence. *Journal of mathematical imaging and vision*, 47(1):60–69, 2013.

Lu Bai, Luca Rossi, Andrea Torsello, and Edwin R Hancock. A quantum jensen–shannon graph kernel for unattributed graphs. *Pattern Recognition*, 48(2):344–355, 2015.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Giorgos Bouritsas, Andreas Loukas, Nikolaos Karalias, and Michael M Bronstein. Partition and code: learning how to compress graphs. *arXiv:2107.01952*, 2021.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *International Conference on Machine Learning*, pp. 1576–1586. PMLR, 2020.

Giovanni Da San Martino, Nicolò Navarin, and Alessandro Sperduti. Tree-based kernel for graphs with continuous attributes. *IEEE transactions on neural networks and learning systems*, 29(7): 3270–3276, 2017.

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

Xinhan Di, Pengqian Yu, Rui Bu, and Mingchao Sun. Mutual information maximization in graph neural networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2020.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in Neural Information Processing Systems*, 32:5723–5733, 2019.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 321–328, 2003.

Kristian Kersting, Nils M Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels. *URL http://graphkernels. cs. tu-dortmund. de*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. In *International Conference on Machine Learning*, pp. 2024–2033. PMLR, 2017.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

Antonio Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: Github as a collaborative social network. In *Eighth international AAAI conference on weblogs and social media*, 2014.

Enxhell Luzhnica, Ben Day, and Pietro Liò. On graph classification networks, datasets and baselines. *arXiv preprint arXiv:1905.04682*, 2019.

Giorgia Minello, Luca Rossi, and Andrea Torsello. Can a quantum walk tell which is which? a study of quantum walk-based graph similarity. *Entropy*, 21(3):328, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4602–4609, 2019.

Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. *Advances in Neural Information Processing Systems*, 33:16211–16222, 2020.

Giannis Nikolentzos, Polykarpos Meladianos, Antoine Jean-Pierre Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. Kernel graph convolutional neural networks. In *International conference on artificial neural networks*, pp. 22–32. Springer, 2018.

Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020.

Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural networks*, 18(8):1093–1110, 2005.

Motakuri V Ramana, Edward R Scheinerman, and Daniel Ullman. Fractional isomorphism of graphs. *Discrete Mathematics*, 132(1-3):247–265, 1994.

Jan Ramon and Thomas Gärtner. Expressivity versus efficiency of graph kernels. In *Proceedings of the first international workshop on mining graphs, trees and sequences*, pp. 65–74, 2003.

Luca Rossi, Andrea Torsello, and Edwin R Hancock. Measuring graph similarity through continuous-time quantum walks and the quantum jensen-shannon divergence. *Physical Review E*, 91(2):022815, 2015.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2018.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Cheng Ye, César H Comin, Thomas K DM Peron, Filipi N Silva, Francisco A Rodrigues, Luciano da F Costa, Andrea Torsello, and Edwin R Hancock. Thermodynamic characterization of networks using graph polynomials. *Physical Review E*, 92(3):032810, 2015.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J Smola. Deep sets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3394–3404, 2017.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

# A APPENDIX

## A.1 EDIT OPERATIONS PROBABILITY OPTIMIZATION

Together with the structural mask $\mathcal{M}$, we optimize also the probabilities $p(e_{i,j})\forall i,j = 1\dots p$ of each edge to belong to the structural mask, and the probability $p(l_i = d)\forall i = 1\dots p, d \in D$ of each node $i$ to assume the label value $d$. Given a structural mask $\mathcal{M} = (L_{\mathcal{M}}, E_{\mathcal{M}})$ with node labels $L = l_1 \dots l_p$ and edge set $E$, the probability distribution of edit operations can be derived from the previous edge and label probabilities as

$$p(E^{'} = E \cup e_{ij}) = \begin{cases} 0 & \text{if } e_{ij} \in E \\ \frac{p(e_{ij})}{\sum_{e_{s,t} \in E} p(e_{s,t}) + \sum_{e_{s,t} \notin E}(1-p(e_{s,t}))} & \text{otherwise} \end{cases} \tag{8}$$

$$p(E^{'} = E - e_{ij}) = \begin{cases} 0 & \text{if } e_{ij} \notin E \\ \frac{1-p(e_{ij})}{\sum_{e_{s,t} \in E} p(e_{s,t}) + \sum_{e_{s,t} \notin E}(1-p(e_{s,t}))} & \text{otherwise} \end{cases} \tag{9}$$

for edge edit operations consisting in either add or remove and edge $e_{i,j}$, while for changing the label $l_i$ of the $i$th node to $d$ as

$$p(l_i^{'} = d, l_j^{'} = l_j \forall i \neq j) = \begin{cases} 0 & \text{if } l_i = d \\ \frac{p(l_i = d)}{\sum_{j, \hat{d} \in D, l_j \neq d} p(l_j = d)} & \text{otherwise} \end{cases} \tag{10}$$

To keep the optimization simpler, we keep two separate distributions for edge and node label editing operations and alternate between the two. The derivatives of edge and node label probabilities are computed according to equation 6 and used as input to standard gradient descent based optimizers (e.g., Adam).

### A.2 Expressive power

**Sketch of proof of Theorem 1** This is trivially the case when we adopt the WL kernel in the GCK layer and the radius $r$ is such that the sub-graphs $N^r_{\mathcal{G}_1}(v)$ span the entire graph $\mathcal{G}_1$ (similarly for $\mathcal{G}_2$). Under this assumption, 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 sub-graphs are identically equal to the graph itself. Assuming that we have a set of structural masks $M = \{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$ 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 GCK layer, the responses of the masks, while identical on all the nodes of each graph, will differ between the two graphs. In particular, the response will peak for any mask fractionally equivalent to the graph itself (Ramana et al., 1994). Therefore, under any reasonable pooling strategy, the GKNN will be able to distinguish between the two graphs.

We now prove that the same holds for an arbitrary sub-graph 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 GCK layer. The prove derives from the fact that the full GCK Layer can be seen as a label propagation step followed by a hashing step. In particular, the computation of the WL kernel against a structural mask, provides the label propagation due to the 1-WL iterations on the sub-graphs; while the implicit dot product against the structural masks, which is in essence a projection onto the space spanned by them, forces a hashing where the adopted masks form a representation space for the hashing itself. Clearly, assuming that we have a sufficiently large (possibly complete) set of structural masks $M = \{\mathcal{M}_1, \cdots, \mathcal{M}_m\}$ that span the space of graphs over $n$ nodes, the dot product induced hashing will not introduce conflicts between nodes with different labels, i.e., labels that are distinct under the 1-WL iterations will eventually be mapped to distinct labels at some GCK layer. Note, however, that under these assumptions the expressive power is greater or equal than that of 1-WL, so much so that labels that are equivalent under 1-WL iteration can indeed be mapped to distinct labels by some GCK 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 sub-graph, which includes cross-neighbor connections. Therefore, given a sufficient number of GCK layers and a reasonable pooling strategy, the GKNN will be able to distinguish between the two input graphs.

**Sketch of proof of Lemma 2** Assume we have a GCK layer of radius 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 sub-graphs of radius 1 with cross-neighbor connections. On these sub-graphs, no star masks will achieve maximum similarity under the WL kernel. Having a pooling process that normalizes the masks responses w.r.t. 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.

Let the radius of the sub-graphs in equation 2 be $r = 1$. Then the sub-graphs 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.

### A.3 SYNTHETIC EXPERIMENTS

**Datasets** For the interpretability analysis experiments we built a dataset for each of the 5 graph motifs. 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, for each Erdős–Rényi graph, two new graphs are built. 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.

**Training details** We trained the network with the same hyper-parameters as in the ablation baseline in section 4.1, with the exception of the number of structural masks, which is set to 8. Moreover, we replaced the ReLU activation of the final MLP with a Sigmoid activation.

### A.4 DATASETS

We validated our approach on a graph classification experiment on the following real-world datasets.

MUTAG (Debnath et al., 1991) 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 (Wale et al., 2008). It has 37 discrete labels.

PROTEINS (Borgwardt et al., 2005; Dobson & Doig, 2003) 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 in 3D space. It has 3 discrete labels, representing helix, sheet or turn. The aim is to distinguish proteins into enzymes and non-enzymes.

PTC (Predictive Toxicology Challenge) dataset (Helma et al., 2001; Kriege & Mutzel, 2012) 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 Borgwardt et al. (2005), in particular 600 enzymes from the BRENDA enzyme database (Schomburg et al., 2004). 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.

IMDB-BINARY dataset (Yanardag & Vishwanathan, 2015) collects graphs about movie collaboration. Each graph consists of nodes representing actors/actress 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.