TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

Theoretical Analysis

We recall that the majority of our work (especially the practical side) focus on using Random Features, mainly OPUs' light-speed random features, to approximate the graphlet kernel to approach Graph Classification problems. We also focus in some stage of this work on using Fourier Random Features to approximate the Gaussian kernel. Thus, in this chapter we first introduce the mathematical notions of Graph Kernels and Random Features. Then, we prove the efficiency of replacing the classical Graph Kernels which have two drawbacks (time, memory) by the use of their corresponding Random Features mapping function, this is to be done using concentration inequalities and information preservation concept. Finally, we show the structure of the OPU and its mathematical model.

1.1 Necessary Notations

Before proceeding to the analysis of used methods, we first introduce the necessary notations related to Graph definition and Graph kernels. A graph by definition is a pair $G = (V, E)$, where V is the set of the graph nodes (vertices) $V = v_1, ..., v_n$, and $E \in V \times V$ is the set of edges between these nodes, i.e. $(v_i, v_j) \in E$ means that the graph has an edge from node v_i to node v_j (or vice versa since we consider undirected graphs in this work).
$H = (V_H, E_H)$ is said to be a subgraph (graphlet) of G ($H \sqsubseteq G$) if and only if there exist an

injective function $\mathcal{M} : V_H \rightarrow V$ such that $(v, w) \in E_H \Leftrightarrow (\mathcal{M}(v), \mathcal{M}(w)) \in E$.

Any edge (v_i, v_i) is called a self loop. In a general graph two vertices v_i and v_j may be connected by more than one edge. A simple graph is a graph with no self loops or multiple edges. Here we always consider simple graphs.

A simple graph can equivalently be represented by an adjacency matrix A of size $n \times n$. The $(i, j) - th$ entry of A is 1 if an edge (v_i, v_j) exists and zero otherwise.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are isomorphic ($G' \cong G$) if there exists a bijective function $\mathcal{M} : V \rightarrow V'$ such that $(v_i, v_j) \in E$ iff $(\mathcal{M}(v_i), \mathcal{M}(v_j)) \in E'$

1.2 Graph Kernels

The kernel trick is a way to generate features for algorithms that depend only on the inner product between pairs of input points. The mathematical basis behind it is that any positive definite function $k(x, y)$ with $x, y \in \mathcal{R}^d$ defines an inner product and a lifting function ϕ so that the inner product between lifted data points equals its kernel $< \phi(x), \phi(y) >= k(x, y)$. The convenience one gets deploying kernels-baed models is that there is no need to access the lifting function ϕ, instead, the used algorithm accesses the data only through evaluations of $k(x, y)$. To better illustrate this benefit, we take the Gaussian Radial Basis function (RBF) kernel as an example, where:

$$K_{RBF}(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2}) \tag{1.1}$$

where σ is called the bandwidth parameter of the kernel. The Hilbert space associated with the Gaussian RBF kernel has infinite dimension, but the kernel may be easily evaluated for any pair (x,y).

1.2.1 Graph kernels design

Traditional kernel machines approach problems with vector-valued input data, where it compare different objects ($x, y \in \mathcal{R}^d$) using the difference between vector components. Based on that, these kernels are imperfect to be used with graphs, since the structure of a graph is invariant to permutations of its representation (e.g. Adjacency matrix), i.e. the ordering by which nodes and edges are enumerated, which leads to isomorphic graphs, does not change the graph structure. So in this case distance-kernels between graph representation vectors are uninformative.As a result it is necessary to measure distance between graphs in ways that are themselves permutation invariant. While the concept of isomorphism is important in learning algorithms on graphs, not only there is no known polynomial-time algorithm for testing graph isomorphism (except for graphs with specific structures), but isomorphism is also too strict for learning in a similar way to learning with equality operator (Kriege, Johansson, and Morris, 2020).

The majority of kernels developed to solve graph learning problems are convolution kernels, where given two discrete structures (e.g., two graphs), the idea of convolution framework is to subdivide these to structures into sub-structures (nodes or subgraphs) and evaluate a kernel between each pair of such sub-structures.

Definition 1 (Convolution Kernel) *let $\mathcal{R} = \mathcal{R}_1 \times ... \times \mathcal{R}_d$ denote a space of components such that a composite object $X \in \mathcal{X}$ decomposes into elements of $\mathcal{R}$. Let $R : \mathcal{R} \to \mathcal{X}$ denote the mapping from components to objects, such that $R(x) = X$ iff the components $x \in \mathcal{R}$ make up the object $X \in \mathcal{X}$, and let $R^{-1}(X) = \{x \in \mathcal{R} : R(x) = X\}$. then, the R-convolution kernel is:*

$$K_{CV}(X, Y) = \sum_{x \in R^{-1}(X)} \sum_{y \in R^{-1}(Y)} \underbrace{\prod_{i=1}^{d} k_i(x_i, y_i)}_{k(x,y)} \tag{1.2}$$

with k_i is a kernel on $\mathcal{R}$ for $i \in \{1, ..., d\}$.

Projecting this definition on Graphs, the inverse map $R^{-1}(G)$ of the convolution kernel

can be seen as the set of all components of a graph G that we want to compare. An example of these kernels is the node label kernel whose mapping R takes the attributes $x_u \in \mathcal{R}$ of each node $u \in G \times H$ and maps them to the graph that u is a member of. The attractive advantage using convolution kernel framework when working with graphs is that while the kernels on substructures are invariant to orderings of nodes and edges, so is the resulting graph kernel. On the other hand, the sum in Eq. 1.2 iterate over all pairs of components. When the considered components become more and more specific, each object becomes increasingly similar to itself, but no longer to any other objects, this problem is called the diagonal dominance problem, since the entries on the main diagonal of the kernel matrix (Gram matrix) are much higher than other entries,Thus, weights between the components usually are added to balance Gram matrix entries so this drawback is resolved.

1.2.2 Graphlet Kernel

Referring by $\mathcal{G} = \{graphlet(1), ..., graphlet(N_k)\}$ to the set of k-nodes graphlets, and considering a graph G of size n we can define a vector f_G of length N_k whose i-th component equals the normalized-number of occurrences of $graphlet(i)$ in G ($\#(graphlet(i) \sqsubseteq G)$. What should be noticed based on this definition is that no two different graphlets in $\mathcal{G}$ are isomorphic. f_G is referred to by k-spectrum of G, and this vector is the key idea behind graphlet kernel.

Definition 2 (Graphlet Kernel) *Given two graphs G and G' of size $n, n' \geq k$, the graphlet kernel k_g is defined as (Shervashidze et al., 2009):*

$$k_g(G, G') = f_G^T f'_G. \tag{1.3}$$

The drawback of this kernel is that computing the k-spectrum vector costs a lot of computational time, since there are $\binom{n}{k}$ size-k subgraphs in a graph G of size n ($O(n^k)$ processing steps required), thus there is a trade off between a more accurate representation

of the graph (large value of k) and the computational cost. However, some techniques are used in order to resolve this limitation as Sampling From Graph technique (section 1.2.3).

1.2.3 Graph Sampling to approximate k-Graphlet Spectrum

The problem of Graph Sampling arises when we deal with a large-scale graph and the task is to pick a small-size sample subgraph/s that would be similar to the original graph with respect to some important properties.

Sampling from graph techniques are used to resolve the processing cost limitation of graphlet kernel, and it can deployed in two different manners:

1. directly sample m sugraphs $\{H_1, ..., H_m\}$ of size k, and then estimate the k-spectrum vector empirically: $f_G(i) = \frac{1}{m}\Sigma_{j=1}^{m}\mathbb{1}[H_j = graphlet(i)]$
2. sample m subgraphs $\{H_1, ..., H_m\}$ of size n' such that $n \gg n' > k$, then estimate f_G as follows: $f_G = \frac{1}{m}\Sigma_{j=1}^{m} f_{H_j}$, this method is usually being referred to by Mean Kernel Method and is used in problems with large-scale or random distribution-based infinity-size graphs (see 1.3.3).

The important thing here is whether a sufficiently large number of random samples will lead to an empirical k-spectrum vector close to the actual vector. The number of samples needed to achieve a given confidence with a small probability of error is called the sample complexity.

Theorem 1 *Let f be a probability distribution on the finite set $\mathcal{G} = g_1, ..., g_a$, $X = X_j{}_{j=1}^{m}$ be a set of independent identically distributed (iid) random variables X_j drawn from f, and $\hat{f}(g_i) = \frac{1}{m}\Sigma_{j=1}^{m}\mathbb{1}(X_j = g_i)$. Then for a given $\epsilon > 0$ and $\delta > 0$ we have (Shervashidze et al., 2009):*

$$m = \left\lceil \frac{2(log(2)a + log(\frac{1}{\delta}))}{\epsilon^2} \right\rceil \tag{1.4}$$

samples suffice to ensure that $P\{\left\| f - \hat{f} \right\|_1 \geq \epsilon\} \leq \delta$

Therefore this theorem gives a lower bound on the number of samples considered respecting the first manner using Graph Sampling to approximate the k-Graphlet spectrum vector in order to ensure some certainty ϵ with prabability $(1 - \delta)$.

1.3 Random Features

Kernel machines are of interest as they approximate any function arbitrarily well with sufficiently large training data set. On the other hand, the methods that operate on the kernel matrix (Gram matrix) require a lot of time in order to compute this matrix and to train the machine; for example, a dataset with half a million training examples might take days to train on modern workstations (Rahimi and Recht, 2008). Unlike kernel machines, linear support vector machines and regularized regression run much faster especially with low-dimensionality training data. One way to combine the advantages of the linear and nonlinear vector machines is to convert the training and evaluation of any kernel machine into the corresponding operations of a linear machine by mapping data into a relatively low-dimensional randomized feature space. Instead of considering the implicit lifting function which corresponds to the kernel, it was proposed to explicitly map the data to a low-dimensional Euclidean inner product space using a randomized feature map $z : \mathcal{R}^d \rightarrow \mathcal{R}^D$ so that the inner product between a pair of transformed points approximates their kernel:

$$k(x, y) =< \phi(x), \phi(y) >\approx z(x)^T z(y) \tag{1.5}$$

Considering this approximation, we can simply transform the input with z and then apply a fast linear learning method to approximate the answer of the real kernel.

In what follows, Random Fourier Features method to construct the random feature map function z is presented.

1.3.1 Random Fourier Features

The following Theorem represents the key idea behind this mapping technique

Theorem 2 (Bochner's theorem) *A continuous and shift-invariant kernel $k(x, y) = k(x-y$ on $\mathcal{R}^d$ is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure.*

What that means is that when a shift-invariant kernel k is properly scaled, its Fourier transform $p(w)$ is a proper probability distribution, we can write:

$$k(x-y) = \int_{\mathcal{R}^d} p(w) e^{jw'(x-y)} dw = E_w[e^{jw'x} e^{jw'y^*}] \tag{1.6}$$

But both $p(w)$ and $k(\delta)$ are real-valued functions, thus from Eq. 1.6 we can prove that:

$$k(x-y) = \int_{\mathcal{R}^d} p(w) cos(w'(x-y)) dw = E_w[z_w(x) z_w(y)] \tag{1.7}$$

where $z_w(x) = \sqrt{2} cos(w'x + b)$ such that w is drawn from $p(w)$ and b is drawn uniformly from $[0, 2\pi]$.

As a straight result, $z_w(x)z_w(y)$ is an unbiased estimate of k(x,y). We can achieve lower variance estimation to the expectation (Eq. 1.7) by averaging D instances of the estimator with different random frequencies w. i.e. the low-variance estimator can be written as: $z(x)'z(y) = \frac{1}{D}\Sigma_{j=1}^{D} z_{w_j}(x) z_{w_j}(y)$. this estimator and based on Hoeffding's inequality guarantees exponentially fast convergence in D between $z(x)'z(y)$ and the kernel true value:

$$Pr(|z(x)'z(y) - k(x,y)| \geq \epsilon) \leq 2e^{\frac{-D\epsilon^2}{4}} \tag{1.8}$$

1.3.2 Random features from compressed sensing's point of view

Random features can be tackled within a different paradigm in signal processing, in compressed sensing for instance a random projection of a high-dimensional but sparse or compressible signal onto a lower-dimensional space has been shown to contain enough information to be used to reconstruct the original signal with small error margins. In this domain,

the well-known Johnson-Lindernstrauss (JL) lemma states that with high probability the geometry of a point cloud is preserved by certain Lipschitz mapping onto a space of dimension logarithmic in the number of points, and with the help of concentration inequalities for random inner products more efficient algorithms for constructing such embeddings were developed. First, we recall ℓ_p^N norm of a vector $x \in \mathcal{R}^N$:

$$\|x\|_{\ell_p^N} = \begin{cases} (\Sigma_{i=1}^N x_i^p)^{\frac{1}{p}} & , 0 < p < \infty \\ max_{i=1,\dots,N} \|x_i\| & , p = \infty \end{cases} \tag{1.9}$$

In the discrete compressed sensing, we want to economically record information about a vector $x \in \mathcal{R}^N$, thus we allocate a group of n nonadaptive questions to ask about x. Each question takes the form of a linear function applied to x, i.e. the extracted information can be written in the form:

$$y = \phi x \tag{1.10}$$

where $\phi \in \mathcal{R}^{n \times N}$ and n is much smaller than N.

To reconstruct the original signal from the information that y holds about x, a decoder Δ is used to provide an approximation $\bar{x} = \Delta(y) = \Delta(\phi x)$. It should be noticed in our case with graph classification problem using random features that it is not one of our concerns to find or prove the proficiency of any decoder, but it is important to cover the mathematics that prove the efficiency of random projections represented in compressed sensing by the pair (ϕ, Δ) . to measure the performance of an encoder-decoder pair (ϕ, Δ), we use a norm $\|.\|_X$ to quantify the error:

$$e(x, \phi, \Delta)_X = \|x - \Delta(\phi x)\|_X \tag{1.11}$$

which is the error of the pair (ϕ, Δ) on x. Moreover, if K is any compact set in $\mathcal{R}^N$, then the error of this pair on K is:

$$e(K, \phi, \Delta)_X = \sup_{x \in K} e(x, \phi, \Delta)_X \tag{1.12}$$

To find the best pair that minimizes the previous error, we introduce the set $\mathcal{A}_{n,N} = \{(\phi, \Delta) : \phi \in \mathcal{R}^{n \times N}\}$, thus the best performance of such pair on K is given by:

$$e_{n,N}(K)_X = \underset{(\phi,\Delta)\in\mathcal{A}_{n,N}}{inf} e(K, \phi, \Delta)_X \tag{1.13}$$

However, it was proven that if $K \subset \mathcal{R}^N$ such that $K = -K$ and that $K + K \subset C_1 K$,where C_1 is a constant, then(Baraniuk et al., 2008):

$$d^n(K)_X < e_{n,N}(K)_X < C_1 d^n(K)_X \tag{1.14}$$

where $d^n(K)_X$ is Gelfand width of K in the Banach space X:

$$d^n(K)_X = \underset{codim(Y)=n}{inf} \underset{x\in K\cap Y}{sup} \|x\|_X \tag{1.15}$$

Here the best spaces Y are those that slice through K in the most economical direction to minimize the diameter of the set $K \cap Y$ (which corresponds to the direction with minimum variance notion used in Principle Component Analysis PCA). An important result of Gelfand widths that can be used in our paradigm is the limits of this width on the unit ball $K = U(\ell_1^N)$, it states that there exists a constant $C_0 > 0$ such that the following condition is satisfied whenever $0 < n < N$:

$$C_0^{-1}\sqrt{\frac{log(N/n)+1}{n}} \leq d^n(U(\ell_1^N))_{\ell_2^N} \leq C_0\sqrt{\frac{log(N/n)+1}{n}} \tag{1.16}$$

Back to the proof of Eq. 1.14 Which can be done by checking the correspondence between Y and ϕ, where given any matrix ϕ, its null space $\mathcal{N}$ is a valid candidate for Y in computing $d^n(K)_X$. On the other hand, given any Y of co-dimension n used in computing $d^n(K)_X$, any basis for the orthogonal complement of Y can be used as the rows of a matrix ϕ to estimate $E_{n,N}(K)_X$. For example, the unit ball $U(\ell_1^N)$ in ℓ_2^N satisfies that $U(\ell_1^N) + U(\ell_1^N) \subset 2U(\ell_1^N)$, thus for all $0 < n < N$:

$$C_0^{-1}\sqrt{\frac{log(N/n)+1}{n}} \leq E_{n,N}(U(\ell_1^N))_{\ell_2^N} \leq 2C_0\sqrt{\frac{log(N/n)+1}{n}} \tag{1.17}$$

The main problem then is to find the pair (ϕ, Δ) which provides estimates like Eq. 1.17, to address this question an isometry condition on ϕ was introduced. Given a matrix ϕ and any set T of column indices, we refer by ϕ_T to the $n \times \#(T)$ matrix composed of these columns. Simultaneously, for every $x \in \mathcal{R}^N$, we refer by x_T to the vector obtained by considering only the entries in x which correspond to the column indices T. We say that a matrix ϕ satisfies the RIP (Ristricted Isometry Property) of order k if there exists a $\delta_k \in [0, 1]$ such that:

$$(1 - \delta_k) \|x_T\|_{\ell_2^N}^2 \leq \|\phi_T x_T\|_{\ell_2^N}^2 \leq (1 + \delta_k) \|x_T\|_{\ell_2^N}^2 \tag{1.18}$$

holds for every such set T with $\#T \leq k$. The good matrices ϕ should satisfy the RIP condition for the largest possible k. For instance, if ϕ satisfies the RIP of order $3k$ with $\delta_{3k} < 1$, then:

$$\|x - \Delta(\phi x)\| \leq \frac{c_2 \sigma_k(x)_{\ell_1^N}}{\sqrt{(}k)} \tag{1.19}$$

where $\sigma_k(x)_{\ell_1^N}$ is the ℓ_1 error of the best k-term approximation and the constant C_2 depends only on δ_{3k}. Since $\sigma_k(x)_{\ell_1^N} \leq \|x\|_{\ell_1^N}$, we can obtain the best possible performance correspondent to Eq.1.17 if we can find a matrix ϕ that meet the RIP condition for k of order $n/(log(N/n) + 1)$.

Now the question is how to construct such matrix ϕ that satisfy the RIP for the largest possible range of k. Actually, the only known constructions yielding such matrices are based on random matrices (Baraniuk et al., 2008). It is shown that matrices built using random entries drawn from certain probability distributions will meet the RIP Condition with high probability. More specifically, for these constructions of ϕ, the RIP follows in a simple way from the same concentration of measure inequalities for inner products that have employed to prove the JL lemma (which will not be stated here being irrelevant).

Briefly, it is shown (Dimitris, 2001) that any random variable which satisfies certain moment conditions, the matrix ϕ whose entries are independent realizations of this variable can be proven to satisfy the following concentration inequality for any $\epsilon \in [0, 1]$:

$$Pr(|\ \|\phi x\|_{\ell_2^N}^2 - \|x\|_{\ell_2^N}^2\ | \geq \epsilon \|x\|_{\ell_2^N}^2) \leq 2e^{-nc_0(\epsilon)} \tag{1.20}$$

Where $c_0(\epsilon)$ is only a function of ϵ. The point now is to prove satisfying Eq.1.20 with some covering arguments is sufficient to prove the RIP for the corresponding matrix ϕ, then some examples of these distributions are presented. Considering the same aforementioned notation ϕ_T, X_T with $\#T \leq k$ with ℓ_2 norm, the proof includes constructing nets of points in each k-dimensional subspace, apply 1.20 to all of these points through a union bound, and then extend the result from our finite set of points to all possible k-dimensional signals.

Lemma 1 *let $\phi(w), w \in \omega^{n\times N}$, be a random matrix of size $n\times N$ drawn from any distribution that satisfies the concentration inequality (1.20). Then, for any set T with $\#T = k < n$ and any $0 < \delta < 1$, we have*

$$(1-\delta)\|x\|_{\ell_2^N} \leq \|\phi(w)x\|_{\ell_2^N} \leq (1_+\delta)\|x\|_{\ell_2^N} \ for\ all\ x \in X_T \tag{1.21}$$

with probability

$$\geq 1 - 2(12/\delta)^k e^{-c_0(\delta/2)n} \tag{1.22}$$

The first step of the proof is noticing that since ϕ represents a linear transformation, it is enough to prove 1.21 in the case $\|x\|_{\ell_2^N} = 1$. The next step is to choose a finite set of points $Q_T \subset X_T$ with $\|q\|_{\ell_2^N} = 1$ for all $q \in Q_T$, that satisfies:

$$\min_{q\in Q_T} \|x-q\|_{\ell_2^N} \leq \delta/4 \ \ for\ all\ x \in X_T \tag{1.23}$$

The existence of such group is proven within the scope of covering numbers where also we can find that $\#Q_T \leq (12/\delta)^k$. Now, using the union bound to apply 1.20 to this Q_T with $\epsilon = \delta/2$, with probability exceeding the right side of 1.22, we have:

$$(1-\delta/2)\|q\|_{\ell_2^N} \leq \|\phi(w)q\|_{\ell_2^N} \leq (1+\delta/2)\|q\|_{\ell_2^N} \ for\ all\ q \in Q_T \tag{1.24}$$

Let A be the smallest number such that:

$$\|\phi(w)x\|_{\ell_2^N} \leq (1+A)\|x\|_{\ell_2^N} \ for\ all\ x \in X_T \tag{1.25}$$

But there is $q \in Q_T$ such that $\|x-q\|_{\ell_2^N} \leq \delta/4$, so we have:

$$\|\phi(w)x\|_{\ell_2^N} \leq \|\phi(w)q\|_{\ell_2^N} + \|\phi(w)(x-q)\|_{\ell_2^N} \leq 1 + \delta/2 + (1+A)\delta/4 \tag{1.26}$$

By a simple comparison between Eq.1.25 and Eq. 1.26, we get that $A \leq \delta$, which is what we seek proving the upper inequality in Eq. 1.21, and the lower one can be proven in a similar way.

Theorem 3 *when n<N, and* $0 < \delta < 1$*, If the probability distribution generating the* $n \times N$ *matrices* $\phi(w), w \in \Omega^{n \times N}$*, satisfies the concentration inequality 1.20, then there exist constants* $c_1, c_2 > 0$ *depending only on* δ *such that the RIP condition holds for* $\phi(w)$ *with the prescribed* δ *and any* $k \leq c_1 n/log(N/k)$ *with probability* $\leq 1 - 2e^{-c_2 n}$*.*

To prove this theorem, from Eq.1.21 we know that the matrix $\phi(w)$ doesn't satisfy the inequality with probability $\leq 2(12/\delta)^k e^{-c_0(\delta/2)n}$ for each of the k-dimensional sub-spaces. But there are $\binom{N}{k} \leq (eN/k)^k$ such sub-spaces. Thus, this probability for all the sub-spaces becomes:

$$\leq 2(eN/k)^k (12/\delta)^k e^{-c_0(\delta/2)n} = 2e^{-c_0(\delta/2)n + k[log(12/\delta) + log((eN/k))]} \tag{1.27}$$

So for a fixed $c_1 > 0$, and $k \leq c_1 n/log(N/k)$, we have that the exponent in Eq. 1.27 is $\leq -c_2 n$ provided that $c_2 \leq c_0(\delta/2) - c_1[1 + (1 + log(12/\delta))/log(N/k)]$. Hence, when $c_1 > 0$ is sufficiently small we have $c_2 > 0$. So this is what we seek, This proves that with probability $1 - 2e^{-c_2 n}$, the matrix $\phi(w)$ will satisfy Eq. 1.21 on all the k-dimensional sub-spaces for the range of $k \leq c_1' n/[log(N/n) + 1]$ for c_1' depending only on the aforementioned c_1.
The main example of such distributions that satisfy the inequality in (1.20) (so the moments of it satisfy the prerequisites of this inequality) is the random matrix ϕ whose entries are independent realizations of Gaussian random variable (Baraniuk et al., 2008):

$$\phi_{i,j} \sim \mathcal{N}(0, \frac{1}{n}) \tag{1.28}$$

Where the corresponding constant c_0 is $c_0(\epsilon) = \epsilon^2/4 - \epsilon^3/6$. Another two examples related

to Bernouli distribution which has the same value of the constant $c_0(\epsilon)$:

$$\phi_{i,j} = \begin{cases} +1/\sqrt{n} \text{ with probability } \frac{1}{2} \\ -1/\sqrt{n} \text{ with probability } \frac{1}{2} \end{cases} \tag{1.29}$$

$$\phi_{i,j} = \begin{cases} +\sqrt{3/n} \text{ with probability } \frac{1}{6} \\ 0 \text{ with probability } \frac{2}{3} \\ -\sqrt{3/n} \text{ with probability } \frac{1}{6} \end{cases} \tag{1.30}$$

1.3.3 Mean kernel and random features

The mean kernel methodology allows to *lift* a kernel from a domain $\mathcal{X}$ to a kernel on *probability distributions* on $\mathcal{X}$. Given a base kernel k and two probability distribution P, Q, it is defined as

$$k(P, Q) = \mathbb{E}_{x \sim P, y \sim Q} k(x, y) \tag{1.31}$$

In other words, the mean kernel is just the expectation of the base kernel with respect to each term. The associated Euclidean metric is referred to by the *Maximum Mean Discrepancy (MMD)*, and is naturally defined as:

$$MMD(P, Q) = \sqrt{k(P, P) + k(Q, Q) - 2k(P, Q)} \tag{1.32}$$

It should be noticed here that $k(P, P) = \mathbb{E}_{x \sim P, x' \sim P} k(x, x') \neq \mathbb{E}_{x \sim P} k(x, x)$. NOw given data points $(x_1, \ldots, x_n)$ drawn *iid* from P and $(y_1, \ldots, y_n)$ drawn *iid* from Q, the kernel in Eq. 1.31 can naturally be approximated by:

$$k(P, Q) \approx \frac{1}{n^2} \sum_{i,j=1}^{n} k(x_i, y_j) \tag{1.33}$$

and the corresponding approximate MMD is (other variants exist):

$$MMD(P, Q) \approx \sqrt{\frac{1}{n^2} \sum_{i,j=1}^{n} k(x_i, x_j) + k(y_i, y_j) - k(x_i, y_j) - k(x_j, y_i)}$$

MMD and random features Mean kernel works especially well with random features. Combining (1.5) and (1.33), it is not hard to see that using random features the mean kernel can be further approximated by:

$$k(P,Q) \approx \frac{1}{n^2}\sum_{i,j=1}^{n}\phi(x_i)^*\phi(y_j) = \left(\frac{1}{n}\sum_i \phi(x_i)\right)^* \left(\frac{1}{n}\sum_i \phi(y_i)\right) \tag{1.34}$$

So the computation can be drastically improved by first computing the *averaged random features* (also called random *generalized moments*, also called *sketch*) $\frac{1}{n}\sum_i \phi(x_i)$, and taking a linear kernel between them. The corresponding MMD is then just the Euclidean metric between the averaged random features

$$MMD(P,Q) \approx \|\frac{1}{n}\sum_i \phi(x_i) - \frac{1}{n}\sum_i \phi(y_i)\|_2$$

MMD for discrete distributions for a discrete space of objects $H_1, \ldots, H_N$ with discrete probability distributions $P = [P_1, \ldots, P_N]$ and Q on them, the mean kernel 1.31 takes a particular form:

$$k(P,Q) = \sum_{i,j=1}^{N} P_i Q_j k(H_i, H_j)$$

MMD with random features on discrete distrbutions To combine both notions One can see the link with graphlet sampling, where f_G is the (discrete) probability distribution of the graphlets. If we define $k(F,F') \approx \phi(F)^*\phi(F')$ where ϕ is a random feature map that replaces ϕ_k, then the feature map (**??**) is exactly what appears in (1.34). So, now, all the game becomes to find a good feature map $\phi(F)$ for graphlets. The induced MMD metric between graphs is the MMD between graphlets probability distributions f_G:

$$d(G,G') = MMD(f_G, f_{G'}) = \sqrt{k(f_G,f_G) + k(f_{G'},f_{G'}) - 2k(f_G,f_{G'})} \approx \|\frac{1}{n}\sum_i \phi(F_i) - \frac{1}{n}\sum_i \phi(F'_i)\|_2$$

where F_i are graphlets drawn from G and F'_i are graphlets drawn from G'.

1.4 Random Projections with Optical Processing Units (OPU's)

Random projections is one of the important techniques in machine learning and signal processing, but it requires either to store a very large random matrix, or to use a different, structured matrix to reduce the computational and memory costs. Optical processing units (OPU's) are the tools used to overcome this difficulty, an OPU performs random projections at the speed of light without having to store any matrix in memory. In general, Random projections are the result of two procedures where the first one is the linear-random projections and the second one is non-linear mapping. Mathematically speaking, OPU's perform the following operation (Saade et al., 2016):

$$X = \phi(WU + b);\ W \in \mathcal{R}^{r \times d}, b \in \mathcal{R}^{r}, U \in \mathcal{R}^{d} \tag{1.35}$$

where W is the random projection matrix, b is a bias, U is an input point, r is the number of random features and d is the input space dimension.With, W is a random i.i.d complex matrix with Gaussian real and imaginary parts and ϕ is the non-linear mapping function.
In the limit where the number of random features $r \to \infty$, it can be proven by the concentration of the measure that the inner product between the projected data points ($X_i \in \mathcal{R}^r$) in the new feature space tends to a kernel function that depends only on the input points in the original feature space ($U_i \in \mathcal{R}^d$)

1.4.1 OPU structure and functionality

Eq. 1.35 still imply that an OPU need to store and multiply by the random projection matrix.

In such units a heterogeneous material, such as paper or any white translucid material, is used to scatter incident light in a very complex way, so the behavior of light scattering is considered random because of the extremely high complexity. one can argue that light

scattering is a linear, deterministic, and reproducible phenomenon, but what can be said is that the unpredictable nature of the process makes it effectively a random process, that is why these materials are called opaque since all information carried within the incident light is seemingly lost during the propagation through the material (Saade et al., 2016). An example used to demonstrate and justify the resulted randomness is a cube of edge length $100\mu m$, such cube can comprise $\approx 10^7$ paint nanoparticles, all the positions and shape of these particles must be known in order to predict its effect on light. Propagation through such a layer can be seen as a random walk because of frequent scattering with the nanoparticles, where light would explore the whole volume and endure on average tens of thousands of such scattering steps before exiting on the other side in a few picoseconds.

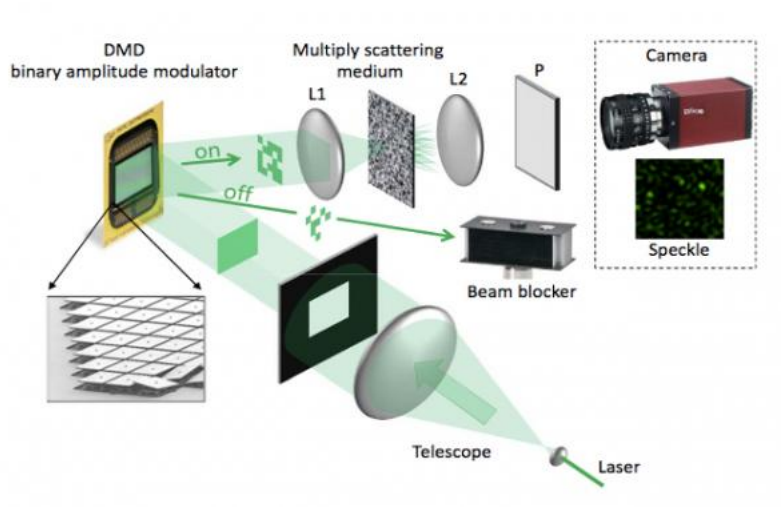

Figure 1.1 OPU's Experimental setup (Saade et al., 2016): A monochromatic laser is expanded by a telescope, then illuminates a digital micromirror device (DMD), able to spatially encode digital information on the light beam by amplitude modulation. The light beam carrying the signal is then focused on a random medium by means of a lens. The transmitted light is collected on the far side by a second lens, passes through a polarizer, and is measured by a standard monochrome CCD camera for example .

When the incident light is coherent, it gives rise to interferences, and the complex interference pattern arising from the multiple scattering process is called speckle. These patterns

are not characteristic only of the propagation medium but also of the shape of the incident light, and this can be modeled by $y = Wx$, where y and x are the vector amplitudes between a set of spatial modes at the output and at the input of the medium respectively and W is the transmission matrix of the medium and it was shown to be very close to Gaussian i.i.d matrices (Saade et al., 2016). What's more convenient to use the aforementioned configuration as a platform for random projections, even without W being determined, is that for a stable medium, such as a paint layer, the transmission matrix W is stable as well. So lightening this layer with an appropriate set of incident illuminations, which is possible using a spatial light modulator and a laser, and measuring the output speckle using CCD or CMOS camera, we can record the resulting intensity $|y|^2$, which is the principle concept behind OPU's functionality as seen in Fig 1.1.

The DMD (digital micromirror device) used in OPU's is a Binary Amplitude Modulator consisting of an array of micro-mirrors, Each encodes a binary value(lit or not). In order to represent grey values, each value is encoded on a square sub-array (4×4 for example) in the DMD,where the number of lit mirrors reflects the desired level of grey. DMD reflects the data through the disordered medium, and a snapshot of the resulting random projection is acquired using a standard camera, all that in a very high speed compared to the traditional random features techniques.

Chapter Two

Results and Discussion

In this chapter we introduce the experiments conducted in this work with a discussion of the results. At first, we present the results of a Synthetic Graph Dataset created based on Stochastic Block Model (SBM), then the results of some real-world datasets (DD, Mutag,..etc).

2.1 Results of Synthetic SBM Dataset

2.1.1 Stochastic Block Model SBM

SBM is commonly known in social sciences to model group structures in friendship graph networks. As a combination of the strict block model with a stochastic element, it was able to deal with imperfect group structures and noise of real world networks. The standard SBM does not only determine the likelihood of a specific group structure belonging to a certain network. The model is based on a generative model, which enables the user to generate other network instances from a given structure or allows the prediction of missing edges (Funke and Becker, 2019).

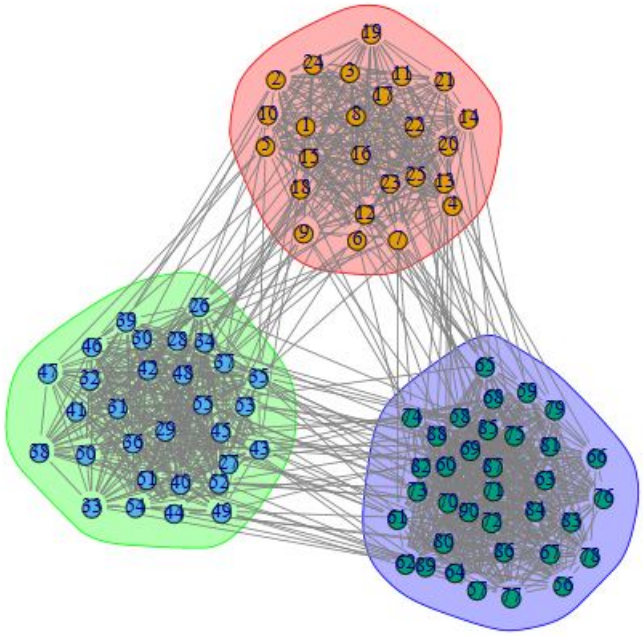

Figure 2.1 An example of a graph generated using SBM model, the graph has 90 nodes divided into three communities of size 25, 30 and 35 nodes. An edge between two nodes within the same community has a probability 0.8, while it has a probability 0.5 if the two nodes belong to different communities.

The basic idea of the standard SBM is that the neighborhood relations of each node only depend on the probabilities assigned to the model. Roughly speaking, the nodes are clustered in a way so that the neighbors of nodes in a group (community) have a similar neighbor pattern as well. To generate a graph G of size n using SBM model, the following parameters should be given: The number of communities (groups) in the graph L, node to community assignments $b_1, \ldots, b_n$ such that node i belongs to community b_i, edge probability matrix $(p_{i,j})_{i,j\in\{1,\ldots,L\}}$. Then the graph is generated by independently add an edge between any two pair of nodes (u, v) with probability p_{b_u,b_v}.

An easy thing to compute here is the average degree d of each node u in the graph $G = SBM(V_G, L, b_u, p_{i,j})$, which is equal to:

$$d_u = \sum_{b\neq b_u} p_{b_u,b} * (\#\{v \in V_G, b_v = b\}) + p_{b_u,b_u} * (\#\{v \in V_G, b_v = b_u\} - 1) \tag{2.1}$$

In our case and almost in every experiment, unless the opposite is mentioned, the dataset

consists of 300 graphs constructed by SBM model each, each graph has $n = 60$ nodes divided equally between two communities $L = 2$. Other parameters $(p_{i,j})$ take different values in different graphs, and actually graphs are divided into different classes based on the corresponding values of these parameters. We consider only the case where the probabilities $p_{1,1} = p_{2,2} = p_{in}$. However, it is obvious that $p_{1,2} = p_{2,1} = p_{out}$ since we want an indirect graphs dataset as indicated previously.

The first class of graphs corresponds to a fixed pair $(p_{in,1}, p_{out,1})$ and similarly the second one corresponds to $(p_{in,2}, p_{out,2})$. These two pairs are always chosen so that any node in any graph in any dataset has an expected average degree equal to 10 (to preserve some difficulty in the classification problem). we refer to $r = (p_{in,1}/p_{in,2})$ by inter-classes similarity parameter, where the closer to one it is the more similar both classes are and thus the harder it is to discriminate them.

2.1.2 Graphlet Kernel results

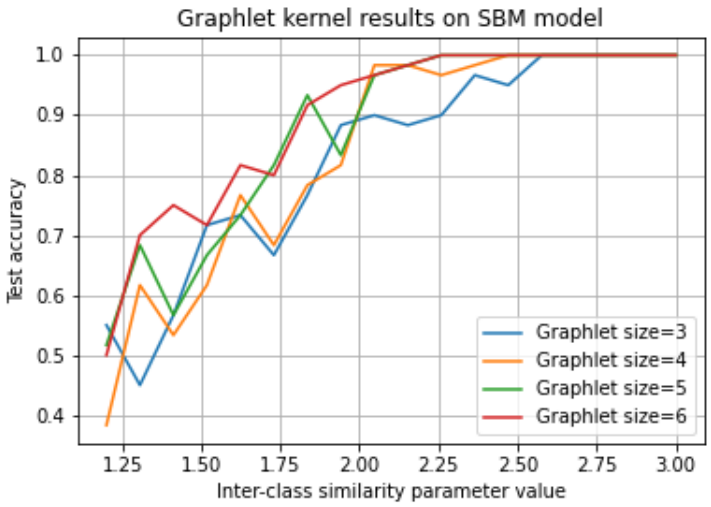

Figure 2.2 Graphlet kernel classification test accuracy with respect to Inter-classes similarity parameter r. where per graph G, 2000 graphlet samples are considered to compute its graphlet spectrum vector.

	3-GRAPHLETS	4-GRAPHLETS	5-GRAPHLETS	6-GRAPHLETS
Computational Time (Sec)	80	120	150	320

Table 2.1 Computational time per epoch of k-graphlet kernel with different k values.

2.1.3 OPUs' Random Features Results

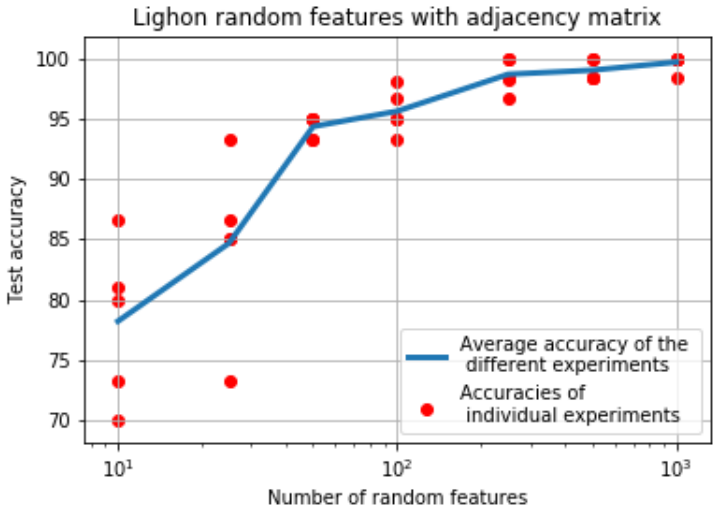

Figure 2.3 Classification test accuracy with respect to the number of LightOn random features. The SVM model is trained on SBM 240-sized labeled dataset. Per graph G, 2000 graphlet samples (Uniform sampling) of size 6 are considered to compute its features map $\phi(G)$. As expected, the accuracy variance drastically decrease as the number of random features increase.

Fixed inter-classes similarity and varying number of random features (Fig. 2.3) The interesting thing here with OPUs is that the computational time does not really depend on the number of random features as long as it is within the capacity of the OPU when we consider fixed graphlet size, in this experiment processing time was as specified in Table. 2.2.

COMPUTATIONAL TIME (SEC)	400 Sec

Table 2.2 Computational time per epoch of OPUs' random features based method.

Graphlet Size	3	4	5
Computational Time (Sec)	630	720	810

Table 2.3 Computational time per epoch of OPUs' random features method using Induced Random walk

Varying inter-classes similarity and fixed number of random features (Fig. 2.4) The computational time in this case is presented in Table. 2.4. We notice that there are slight differences between time values when the graphlet size change, this is due to two reasons, the first one is that OPUs' processing time does not depend on the dimension of the input nor on the number of random features as we respect its capacity, the second one is that we consider Uniform Sampling technique to sub-sample k-size graphlets, which is unlike Random Walk based sampling techniques fast and doesn't require significantly larger time when the graphlet size increases.

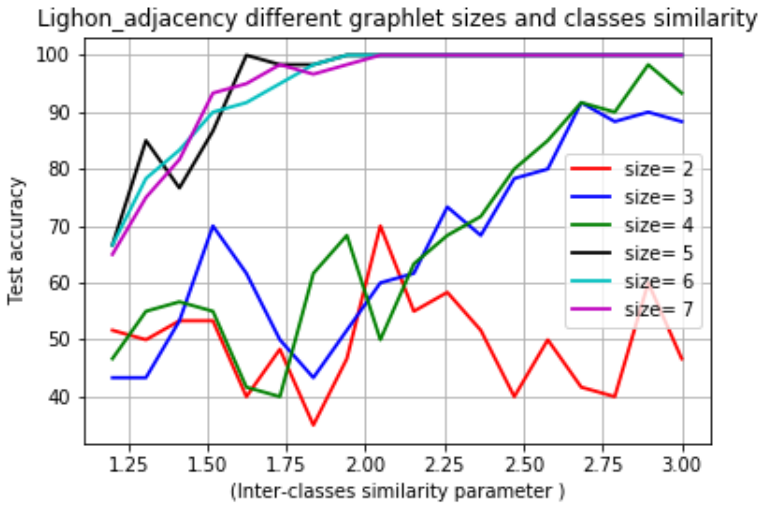

Figure 2.4 Classification test accuracy with respect to Inter-classes similarity parameter when the number of random features is fixed to 5000 but with different sizes of the graphlet to be sampled. The SVM model is trained on SBM 240-sized labeled dataset. Per graph G, 2000 graphlet samples (Uniform sampling) of the corresponding size are considered to compute its features map $\phi(G)$.

Graphlet Size	2	3	4	5	6	7
Computational Time (Sec)	340	357	362	378	400	412

Table 2.4 Computational time per epoch of OPUs' random features based method using uniform sampling technique.

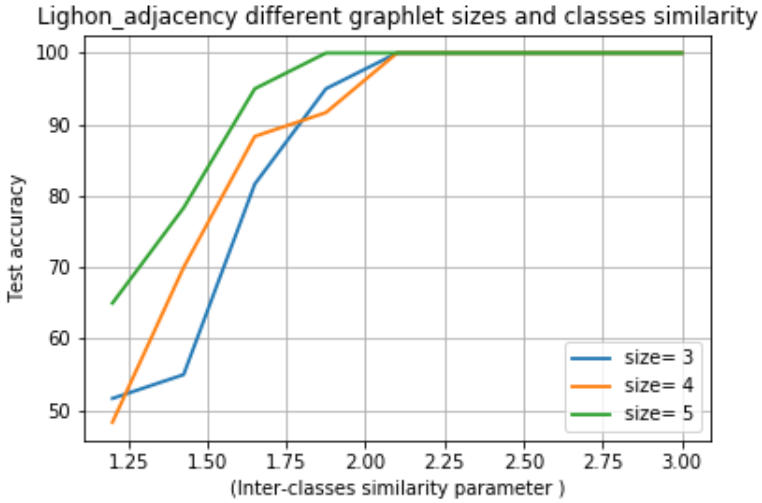

Figure 2.5 Classification test accuracy with respect to Inter-classes similarity parameter when the number of random features is fixed to 5000 but with different sizes of the graphlet to be sampled. The SVM model is trained on SBM 240-sized labeled dataset. Per graph G, 2000 graphlet samples (Random Walk sampling) of the corresponding size are considered to compute its features map $\phi(G)$. This experiment is done to check if the uniform sampling technique is the reason behind the gap between accuracy curves of graphlet sizes 4 and 5 in Fig. 2.4

However, one unexpected thing in Fig. 2.4 is that there is notably a gap between 4-graphlet and 5-graphlet curves, to detect the reason behind this gap we ran the same experiment with the same settings but using Induced Random Walk Sampling Technique , which tends to sample nodes that appear in a random walk starting from randomly chosen node (will be further explained in the report later). results are shown in Fig. 2.5 and the corresponding computational time is shown in Table. 2.5.

Epoch processing time (Sec)	total training time (Sec)
0.29	87

Table 2.5 Computational time per epoch of the GCN GIN-based model.

Graph Convolution Network GCN In order to benchmark the results of Random Features-based methods with other common methods used in Graph classification, we modified and trained one of the proposed models built based on GIN (Graph Isomorphism Network), which has been reported to give brilliant results (even better than most of state-of-art classical Graph Convolutional Networks) in classifying graphs based only on their structural information in the absence of node features just like the case of our SBM dataset (Xu et al., 2018).

Figure 2.6 GCN model's classification test accuracy with respect to Inter-classes similarity parameter. The model is trained on SBM 240-sized labeled dataset.

The model consists of 5 GIN layers followed by two fully connected layers where the dimensions of hidden layers are equal to 4. Comparing GCN performance to the one we got using OPUs random features (Fig. 2.5 and Fig. 2.4) we can say that the performance of Random Features based methods is slightly better when the graphlet size is greater than 4,

especially using Induced Random Walk sampling technique. However, the training time of GCN model still notably better (I should check if I can do something with LightOn platform to enhance its time, so the report will be fixed later).

2.2 DD Real-world Data set

2.2.1 OPUs' Random Features Results

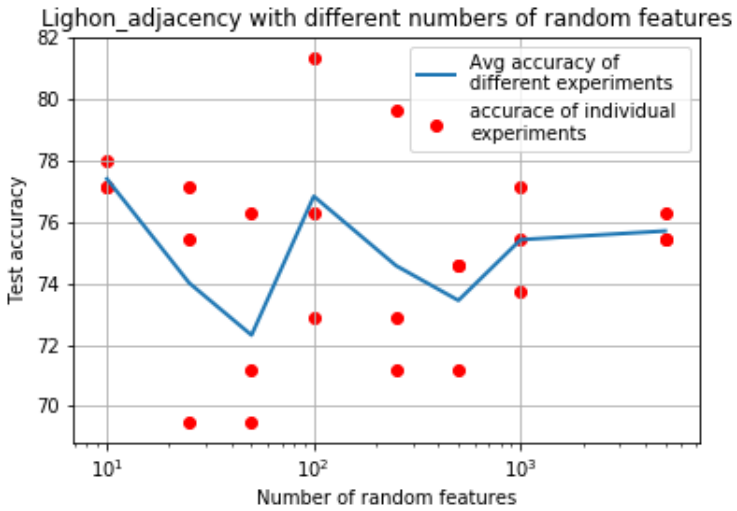

Figure 2.7 Classification test accuracy with respect to the number of random features on DD Dataset

REFERENCES

Baraniuk, Richard et al. (2008). "A simple proof of the restricted isometry property for random matrices". In: *Constructive Approximation* 28.3, pp. 253–263.

Dimitris, Achlioptas (2001). "Database-friendly random projections". In: *Proceedings of the twentieth ACM SIGMOD-SIGACTSIGART symposium on Principles of database systems, PODS'01 New York, NY, USA*. ACM, pp. 274–281.

Funke, Thorben and Till Becker (2019). "Stochastic block models: A comparison of variants and inference methods". In: *PloS one* 14.4.

Kriege, Nils M, Fredrik D Johansson, and Christopher Morris (2020). "A survey on graph kernels". In: *Applied Network Science* 5.1, pp. 1–42.

Rahimi, Ali and Benjamin Recht (2008). "Random features for large-scale kernel machines". In: *Advances in neural information processing systems*, pp. 1177–1184.

Saade, Alaa et al. (2016). "Random projections through multiple optical scattering: Approximating kernels at the speed of light". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6215–6219.

Shervashidze, Nino et al. (2009). "Efficient graphlet kernels for large graph comparison". In: *Artificial Intelligence and Statistics*, pp. 488–495.

Xu, Keyulu et al. (2018). "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826*.