

FAST GRAPH KERNEL WITH OPTICAL RANDOM FEATURES

Hashem Ghanem Nicolas Keriven Nicolas Tremblay

CNRS, GIPSA-lab, FR-38402 Saint Martin d’Heres Cedex, France

ABSTRACT

The graphlet kernel is a classical method in graph classification. It however suffers from a high computation cost due to the isomorphism test it includes. As a generic proxy, and in general at the cost of losing some information, this test can be efficiently replaced by a user-defined mapping that computes various graph characteristics. In this paper, we propose to leverage *kernel random features* within the graphlet framework, and establish a theoretical link with a mean kernel metric. If, at first glance, this method does not necessarily allow to reduce the computational complexity of the graphlet kernel **why do you say that? Isomorphism test is in exponential time; even without the OPU, the proposition of this paper is polynomial, so infinitely faster, no?**, we then incorporate *optical* random features that can compute such random features in *constant time*. Experiments show that the resulting algorithm is orders of magnitude faster than the graphlet kernel for the same, or better, accuracy.

Index Terms— Optical random features, Graph kernels

1. INTRODUCTION

In mathematics and data science, graphs are used to model a set of objects (the *nodes*) and their interactions (the *edges*). Given a set of pre-labeled graphs ($\mathcal{X} = \{\mathcal{G}_1, \dots, \mathcal{G}_n\}$, $\mathcal{Y} = \{y_1, \dots, y_n\}$), where each graph \mathcal{G}_i belongs to the class with label y_i , graph classification consists in designing an algorithm that outputs the class label of a new graph. For instance, proteins can be modeled as graphs: amino acids are nodes and the chemical links between them are edges. They can be classified to enzymes and non-enzymes [1]. In social networks analysis, post threads can be modeled with graphs whose nodes are users and edges are replies to others’ comment [2]. One task is then to discriminate between discussion-based and question/answer-based threads [3]. In addition to the graph structure, nodes and edges may have extra features. While it has been shown that node features are important to obtain high classification performance [4], here we focus on the case where one has only access to the graph structure.

Structure-based graph classification has been tackled with many algorithms. Frequent subgraphs based algorithms [5] analyze the graph dataset \mathcal{X} to catch the frequent and discriminative subgraphs and use them as features. Kernel-based al-

gorithms [6] can be used by defining similarity functions (kernels) between graphs. An early and popular example is the *graphlet kernel*, which computes frequencies of subgraphs. It is however known to be quite costly to compute [7], in particular due to the presence of graph isomorphism tests. While possible in particular cases [?], accelerating the graphlet kernel for arbitrary datasets remains open. Finally, Graph neural networks (GNNs) [?, ?] have recently become very popular in graph machine learning. They are however known to exhibit limited performance when node features are unavailable [?].

In kernel methods, random features are an efficient method to approximate certain kernel functions [?, 8]. Recently, it has been shown*ref* that *optical computing* can be leveraged to compute such random features in *constant time* in *any dimension* – within the limitations of the current hardware, here referred to as Optical Processing Units (OPUs). The main goal of this paper is to provide a proof-of-concept answer to the following question: can OPU computations be used to reduce the computational complexity of a combinatorial problem like the graphlet kernel? Drawing on a connection with mean kernels and Maximum Mean Discrepancy (MMD) [?], we show, empirically and theoretically, that a fast and efficient graph classifier can indeed be obtained with OPU computations.

* the biblio file is not up to date apparently *

* + good job with the intro! *

2. BACKGROUND

First, we present the concepts necessary to define the graphlet kernel. We represent a graph of size v by the adjacency matrix $\mathbf{A} \in \{0, 1\}^{v \times v}$, such that $a_{i,j} = 1$ if there is an edge between nodes $\{i, j\}$ and 0 otherwise. Two graphs are said to be isomorphic ($\mathcal{G} \cong \mathcal{G}'$) if we can permute the nodes’ labels of one such that their adjacency matrices are equal [9].

2.1. Isomorphic graphlets

In this paper, we will, depending on the context, manipulate two different notions of k -graphlets (that is, small graphs of size k), with or without discriminating isomorphic graphlets. We denote by $\bar{\mathcal{H}} = \{\bar{\mathcal{H}}_1, \dots, \bar{\mathcal{H}}_{\bar{N}_k}\}$ with $\bar{N}_k = 2^{\frac{k(k-1)}{2}}$ the set of all size- k graphs, where isomorphic graphs are counted multiple times, and $\mathcal{H} = \{\mathcal{H}_1, \dots, \mathcal{H}_{N_k}\} \subset \bar{\mathcal{H}}$ the set of all

non-isomorphic graphs of size k . Its size N_k has a (quite verbose) closed-form expression [?], but is still exponential in k . In the course of this paper, we shall manipulate mappings $\varphi(\mathcal{H})$ and probability distributions (histograms) over graphlets. When the mapping φ is *permutation-invariant*, then the underlying space is \mathfrak{H} , otherwise it is $\tilde{\mathfrak{H}}$. Using one or the other obeys a tradeoff: $\tilde{\mathfrak{H}}$ is larger, while \mathfrak{H} may necessitate costly isomorphism tests. In any case, assuming each isomorphic copies has equal probability, a probability distribution over $\tilde{\mathfrak{H}}$ can easily be transformed into one over \mathfrak{H} , and both versions contains the same amount of information.

2.2. The graphlet kernel

The traditional graphlet kernel is defined as follows. We define the matching function $\varphi_k^{match}(\mathcal{F}) = [1_{(\mathcal{F} \cong \mathcal{H}_i)}]_{i=1}^{N_k} \in \{0, 1\}^{N_k}$, where 1_Ω is the indicator function and \mathcal{F} is a graph of size k . In words, $\varphi_k^{match}(\mathcal{F})$ is a one-hot vector of dimension N_k identifying \mathcal{F} up to isomorphism. Note that the cost of evaluating φ_k^{match} once is $O(N_k C_k^\cong)$, where C_k^\cong is the cost of the isomorphism test between two graphs of size k , for which no polynomial algorithm is known [10].

Given a graph \mathcal{G} , let $\mathfrak{F}_{\mathcal{G}} = \{\mathcal{F}_1, \mathcal{F}_2, \dots\}$ be the set of subgraphs induced by all size- k subsets of nodes¹. The following representation vector is called the k -spectrum of \mathcal{G} :

$$\mathbf{f}_{\mathcal{G}} = \frac{1}{|\mathfrak{F}_{\mathcal{G}}|} \sum_{\mathcal{F} \in \mathfrak{F}_{\mathcal{G}}} \varphi_k^{match}(\mathcal{F}) \in \mathbb{R}^{N_k} \quad (1)$$

For two graphs $\mathcal{G}, \mathcal{G}'$, the graphlet kernel is defined as $\mathbf{f}_{\mathcal{G}}^T \mathbf{f}_{\mathcal{G}'}$. For a graph of size v , the computation cost of $\mathbf{f}_{\mathcal{G}}$ is $C_{gk} = O\left(\binom{v}{k} N_k C_k^\cong\right)$. This cost is usually prohibitively expensive, since each three terms are exponential in k .

Subgraph sampling is generally used as a first step to accelerate (and sometimes modify) the graphlet kernel [7]. Given a graph \mathcal{G} , we denote by $S_k(\mathcal{G})$ a sampling process that yields a random subgraph of \mathcal{G} , of which there exists many variants [?]. Then, sampling s iid subgraphs $\{F_1, \dots, F_s\}$ of size k from $S_k(\mathcal{G})$, we define the estimator:

$$\hat{\mathbf{f}}_{\mathcal{G}, S_k} = \frac{1}{s} \sum_{F \in \mathfrak{F}_{\mathcal{G}}} \varphi_k^{match}(F). \quad (2)$$

*you sum over $\hat{\mathfrak{F}}_{\mathcal{G}}$ that you have not defined. Do we really need the notation $\hat{\mathfrak{F}}_{\mathcal{G}}$? Perhaps we can say we sum over $\{F_1, \dots, F_s\}$ * and its expectation $\mathbf{f}_{\mathcal{G}, S} = \mathbb{E}_{F \sim S_k(\mathcal{G})} \varphi_k^{match}(F)$, which is a probability distribution over the set of graphlets \mathfrak{H} . We refer to these expectations as *graphlet kernels*. In fact, in all generality, any choice of sampling procedure S_k yields a different definition of graphlet kernel. For instance, if one considers uniform sampling (S^{unif} : independently samples k

nodes of \mathcal{G} without replacement), then one obtains the original graphlet kernel of Eq. (1): $\mathbf{f}_{\mathcal{G}, S^{\text{unif}}} = \mathbf{f}_{\mathcal{G}}$. Other choices of sampling procedures are possible [11]. In this paper, we will also use the random walk (RW) sampler, which, unlike uniform sampling, tends to sample connected subgraphs, which may be more informative about the graph structure.

The computation cost per graph of the *approximate graphlet kernel with graph sampling* of Eq. (2) is $C_{gk+gs} = O(s C_S N_k C_k^\cong)$, where C_S is the cost of sampling one subgraph. For a fixed error in estimating $\mathbf{f}_{\mathcal{G}, S}$, the required number of samples s is generally proportional to N_k *rather needs to be prop to N_k , no?* [7], which *unfortunately still yields an unaffordable algorithm*.

3. GRAPHLET KERNEL WITH OPTICAL MAPS

3.1. Proposed algorithm

Algorithm 1: GSA- φ generic algorithm

Input: labeled graph dataset $\mathcal{X} = (\mathcal{G}_i, y_i)_{i=1, \dots, n}$

- 1 **Tools** Graphlet sampler S_k , a function φ , linear classifier (ex. SVM)
- 2 **Hyperparameters** k : graphlet size, s : number of graphlet samples per graph, m : number of random features

Output: Trained model to classify graphs

- 3 **Algorithm**
 - 4 Random initialization of the SVM weights
 - 5 **for** \mathcal{G}_i in \mathcal{X} **do**
 - 6 $\mathbf{z}_i = \mathbf{0}$ (null vector of size m)
 - 7 **for** $j = 1 : s$ **do**
 - 8 $F_{i,j} \leftarrow S_k(\mathcal{G}_i)$
 - 9 $\mathbf{z}_i \leftarrow \mathbf{z}_i + \frac{1}{s} \varphi(F_{i,j})$
 - 10 $\mathcal{D}_{\varphi} \leftarrow (\mathbf{z}_i, y_i)_{i=1, \dots, n}$
 - 11 Train the linear classifier on the new vector-valued dataset \mathcal{D}_{φ}
-

In this paper, we propose to deal with the main remaining bottleneck of the graphlet kernel, that is, the function φ_k^{match} . First, we therefore define a framework where it is replaced with a user-defined map $\varphi : \mathfrak{H} \rightarrow \mathbb{R}^m$. The resulting algorithm (Alg. 1) is referred to as *graphlet sampling and Averaging* (GSA- φ). The cost of computing $\mathbf{z}_{\mathcal{G}}$ is dominated by $C_{GSA-\varphi} = O(s C_S C_{\varphi})$, where C_{φ} is the cost of applying φ on one graphlet. Note that choosing $\varphi = \varphi_k^{match}$ and S_k as the uniform sampler, GSA- φ_k^{match} turns out to be the graphlet kernel with sampling. We see next that choosing φ as *kernel random maps* is both fast and efficient. Some of these maps will *not* be permutation-invariant at the graphlet level, however, in the infinite sample limit, it is easy to see that the representation vector $\mathbf{z}_{\mathcal{G}}$ is indeed permutation-invariant at the graph level.

¹ $|\mathfrak{F}_{\mathcal{G}}|$ depends on \mathcal{G} and is usually much smaller than N_k . Extreme examples are the complete and the empty graphs, where $|\mathfrak{F}_{\mathcal{G}}| = 1$.

3.2. Efficiency of kernel random features with $GSA - \varphi$

By averaging φ_k^{match} , the graphlet kernel obtains discrete probability distributions $\hat{\mathbf{f}}_{\mathcal{G}, S_k}$. The underlying metric used to compare graphs is the Euclidean distance between these histograms. When φ_k^{match} is replaced by another φ , one compare Euclidean distance between certain *embeddings* of distributions, which is reminiscent of kernel mean embeddings [?]. We show below that this corresponds to choosing φ as kernel random features.

For two objects \mathbf{x}, \mathbf{x}' , a kernel κ associated to a random features (RF) decomposition is a positive definite function that can be decomposed as follows [12]:

$$\kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w} \sim p} [\xi_{\mathbf{w}}(\mathbf{x})^* \xi_{\mathbf{w}}(\mathbf{x}')] \quad (3)$$

where ξ is a real (or complex) function parameterized by $\mathbf{w} \in \Omega$, and p is a probability distribution on Ω . Classical examples include Fourier decomposition of translation-invariant kernels [12]. The RF methodology then defines maps:

$$\varphi(\mathbf{x}) = \frac{1}{\sqrt{m}} (\xi_{\mathbf{w}_j}(\mathbf{x}))_{j=1}^m \in \mathbb{C}^m \quad (4)$$

where m is the number of features and the parameters \mathbf{w}_j are drawn iid from p . Then, $\kappa(\mathbf{x}, \mathbf{x}') \approx \varphi(\mathbf{x})^H \varphi(\mathbf{x}') = \frac{1}{m} \sum_j \xi_{\mathbf{w}_j}(\mathbf{x})^* \xi_{\mathbf{w}_j}(\mathbf{x}')$.

Assume that we have a base kernel $\kappa(\mathcal{F}, \mathcal{F}')$ between *graphlets*, with a RF decomposition $(\xi_{\mathbf{w}}, p)$, and define φ as in (4). Then, it is not hard to see that the Euclidean distance between the embeddings computed by Alg. 1 $\|\mathbf{z}_{\mathcal{G}} - \mathbf{z}_{\mathcal{G}'}\|_2^2$ approximates the following *Maximum Mean Discrepancy* (MMD) [?, ?] between the two discrete distributions of graphlets :

$$\begin{aligned} &MMD^2(S_k(\mathcal{G}), S_k(\mathcal{G}')) \\ &= \mathbb{E}_{\mathbf{w}} \left(\left| \mathbb{E}_{S_k(\mathcal{G})} \xi_{\mathbf{w}}(\mathcal{F}) - \mathbb{E}_{S_k(\mathcal{G}')} \xi_{\mathbf{w}}(\mathcal{F}') \right|^2 \right) \end{aligned} \quad (5)$$

Note that here, $S_k(\mathcal{G})$ can be interpreted as a distribution over \mathfrak{H} if $\xi_{\mathbf{w}}$ is permutation-invariant, and over $\bar{\mathfrak{H}}$ otherwise. The main property of the MMD is that, for so-called *characteristic kernels*, it is a true metric on distributions, *i.e.* $MMD(S_k(\mathcal{G}), S_k(\mathcal{G}')) = 0 \Leftrightarrow S_k(\mathcal{G}) = S_k(\mathcal{G}')$, that is, it is 0 if and only if \mathcal{G} and \mathcal{G}' incur the same distribution over graphlets. In addition, most usual kernels, like the Gaussian kernel, are characteristic [13]. The next theorem, whose proof can be found in the supplementary material, bounds the quality of the approximation.

todo: S_k as a distribution over non-isomorphic graphlets. Isomorphism is a special case

Theorem 1. *Let \mathcal{G} and \mathcal{G}' be two graphs. Assume a random feature map as in (4). Assume that $|\xi_{\mathbf{w}}(F)| \leq 1$ for any \mathbf{w}, F .*

We have for all $\delta > 0$ and with probability at least $1 - \delta$:

$$\begin{aligned} &\left| \|\mathbf{z}_{\mathcal{G}} - \mathbf{z}_{\mathcal{G}'}\|_2^2 - MMD^2(S_k(\mathcal{G}), S_k(\mathcal{G}')) \right| \\ &\leq \frac{4\sqrt{\log(6/\delta)}}{\sqrt{m}} + \frac{8 \left(1 + \sqrt{2\log(3/\delta)} \right)}{\sqrt{s}} \end{aligned}$$

According to this result, the number of random features m should approximately be of the order of the number of samples s . We have seen that the latter is usually quite high, hence most usual random feature scheme will still incur a high computation cost overall. We discuss next random features with efficient optical hardware.

3.3. Considered choices of φ_{RF}

Gaussian maps φ_{Gs} : the RF map of the Gaussian kernel [12]. For a graphlet \mathcal{F} we define its vectorized adjacency matrix $\mathbf{a}_{\mathcal{F}} = \text{flatten}(\mathbf{A}_{\mathcal{F}})$, then:

$$\varphi_{Gs}(\mathcal{F}) = m^{-1/2} \left(\sqrt{2} \cos(\mathbf{w}_j^T \mathbf{a}_{\mathcal{F}} + b_j) \right)_{j=1}^m \in \mathbb{R}^m \quad (6)$$

where the $\mathbf{w}_j \in \mathbb{R}^{k^2}$ are drawn from a Gaussian distribution and $b_j \sim \mathcal{U}([0, 2\pi])$. While using a Gaussian kernel on $\mathbf{a}_{\mathcal{F}}$ is not necessarily intuitive, this will serve as a baseline for the other methods. Note that φ_{Gs} is not permutation-invariant.

Gaussian maps φ_{Gs+Eig} applied on the sorted eigenvalues: We consider a permutation-invariant alternative to the first case. For a graphlet \mathcal{F} we denote the vector of its sorted eigenvalues by $\lambda(\mathcal{F}) \in \mathbb{R}^k$ and $\varphi_{Gs+eig}(\mathcal{F}) = \varphi_{Gs}(\lambda(\mathcal{F}))$ (with \mathbf{w}_j of dimension k). Note that the existence of co-spectral graphs, that is, non-isomorphic graphs with the same set of eigenvalues, implies a loss of information when computing $\lambda(\mathcal{F})$.

Optical random feature maps φ_{OPU} : Due to high-dimensional matrix multiplication, Gaussian RF cost $\mathcal{O}(mk^2)$ and are notably expensive to compute in high-dimension (in this case, high m in particular). To solve this, OPUs (Optical Processing Units) technology were recently developed to compute a specific random features mapping in *constant time* $\mathcal{O}(1)$ in both m and k using light scattering [14] – within the current physical limitations of the devices, actually of the order of a few millions for both input and output dimensions. Here we again consider the flattened adjacency matrix for simplicity, and the corresponding map is:

$$\varphi_{OPU}(\mathbf{A}_{\mathcal{F}}) = |\mathbf{W}\mathbf{A}_{\mathcal{F}} + \mathbf{b}|^2; \quad \mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^d$$

Where \mathbf{b} is a random bias vector, d is the input space dimension, the amplitude function $|\cdot|$ is taken element wise, and \mathbf{W} is a random *iid* complex matrix with Gaussian real and imaginary parts.

The complexities of the different mappings φ examined in this work are summarized in Table 1.

Graphlet kernel		$O(\binom{v}{k} N_k C_k^{\frac{s}{k}})$
GSA- φ with:	φ_k^{match}	$O(C_s s N_k C_k^{\frac{s}{k}})$
	φ_{Gs}	$O(C_s s m k^2)$
	$\varphi_{Gs+Eigen}$	$O(C_s s (m k + k^3))$
	φ_{OPU}	$O(C_s s)$

Table 1. Per-graph complexities of GSA- φ .

4. EXPERIMENTS

4.1. Setup

With respect to the performance and the computation cost, we compare different choices of map φ in $GSA - \varphi$. We benchmark the performance of $GSA - \varphi_{OPU}$ against GIN based graph convolutional network.

In all experiments except the last one, we use a synthetic dataset generated by a *Stochastic Block Model (SBM)* [15]. We generate 300 graphs, 240 for training and 60 for testing. Each graph has $v = 60$ nodes divided equally between six communities. Moreover, graphs are divided into two classes $\{0, 1\}$ based on the edges distribution considered. For each class we fix two values (p_{in}, p_{out}) which are the probabilities of generating an edge between any two nodes when they are in the same community and when they are in different ones, respectively. Besides, to prevent the classes from being easily discriminated by the average degree, the pairs $(p_{in,i}, p_{out,i})_{i=0,1}$ are chosen such that all nodes have a fixed expected average degree equal to 10. Having one degree of freedom left, we fix $p_{in,1} = 0.3$, and we vary $r = (p_{in,1}/p_{in,0})$ the inter-class similarity parameter: the closer r is to 1, the more similar both classes are, and thus the harder it is to discriminate them.

On the other hand, D&D is a labeled dataset of size $n = 1178$ protein graphs [16]. Also, nodes have 7 features each, which are not used by our algorithms, *i.e.* we will try to classify the graphs just based on their structure. In what follows, unless otherwise indicated, we use uniform sampling and the adjacency matrix of subgraphs as input.

4.2. Choice of feature map φ

Comparison of random features: Fig 1(a) shows that $GSA - \varphi_{OPU}$ applied on adjacency matrices gives better test accuracy with sufficiently large m than both $GSA - \varphi_{Gs}$ applied on adjacency matrices or $GSA - \varphi_{Gs+Eig}$ applied on its sorted eigenvalues. On the contrary, $GSA - \varphi_{Gs+Eig}$ performs best with a low number of random features, but increasing this number does not really improve the result and it is over-matched at high m . A possible justification is that the eigenvalues of the adjacency matrix lose information about the subgraphs, even though respecting the isomorphism means that we are working with a smaller histogram and less random features are required.

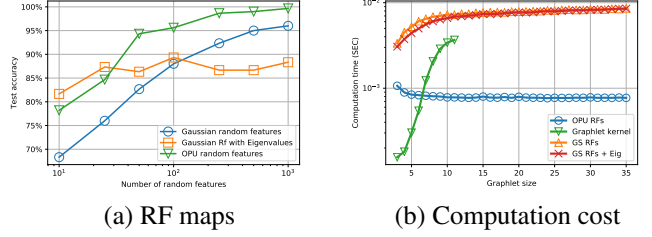


Fig. 1. Comparing different φ maps in $GSA - \varphi$. (a) test accuracy when using RF maps with $k = 6$ while varying m . (b) test accuracy of $GSA - \varphi_{OPU}$ and $GSA - \varphi_k^{match}$ fixing $k = 5$ while varying r . (c) Computation time as a function of k . If not specified: $r = 1.1$, $s = 2000$, $m = 5000$ and the Gaussian map variance $\sigma^2 = 0.01$.

Comparing $GSA - \varphi_{OPU}$ to $GSA - \varphi_k^{match}$: from Fig 1(b) we observe that with the same limited number of samples s , $GSA - \varphi_{OPU}$ clearly outperforms the approximated graphlet kernel $GSA - \varphi_k^{match}$, so we conclude that $GSA - \varphi_{OPU}$ is more adapted in this case than the traditional graphlet kernel.

Computational time: Fig 1(c) shows the computation time of each of the previous methods with respect to the subgraphs size k . Other parameters are identically fixed for all methods. As expected, the execution time of the approximated graphlet kernel grows exponentially with k , and is polynomial for $GSA - \varphi_{Gs}$ and $GSA - \varphi_{Gs+Eig}$. On the contrary, it is almost constant for $GSA - \varphi_{OPU}$. We point out here that, with the current settings, there is a significant overhead between the point in time where we run our optimizer code on the OPUs server and the point when the OPU launches the computation. To be fair, this overhead time should be measured and subtracted from $GSA - \varphi_{OPU}$ computation time. As the technology comes into maturity, we can expect this additional time to be significantly reduced.

To summarize, $GSA - \varphi_{OPU}$ outperforms the traditional methods both in accuracy and computational cost.

4.3. Comparing $GSA - \varphi_{OPU}$ against GIN model

In Fig 2, we observe that for subgraphs sizes > 5 , both RW and uniform sampling perform similarly well in $GSA - \varphi_{OPU}$, but RW sampling, as expected, provide more consistent results when the graphlet size k varies. Thus RW sampling is considered in this comparison. We see that $GSA - \varphi_{OPU}$ with RW performs better than the GIN model when the graphlet size is greater than 4. We note that we do not report the computational time for GIN, since it is highly dependent on high-speed graphical processing units (GPUs) to do the training process.

In Fig 3, we have the test accuracy with varying value of m and fixed $s = 4000$, $k = 7$. For each value of m we conduct the experiment 5 times and take the average accu-

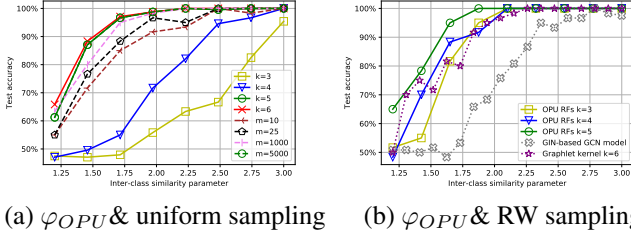


Fig. 2. Comparing test accuracies of $GSA - \varphi_{OPU}$ and GIN-based GCN network when varying the problem difficulty r . We used $GSA - \varphi_{OPU}$ with uniform sampling in (a) and with random walk sampling in (b). In both cases: $s = 2000$ and $m = 5000$. (c) The model consists of 5 GIN layers then 2 fully connected layers, the dimensions of hidden layers: 4.

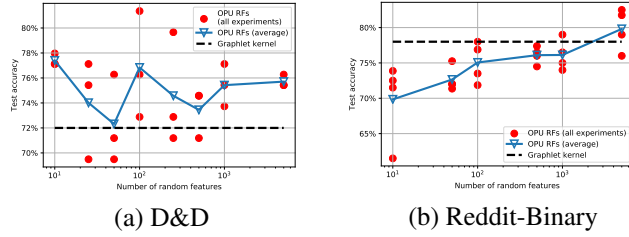


Fig. 3. Bench-marking $GSA - \varphi$ against the graphlet kernel as a performance reference on real datasets with: $s=4000, k=$.

racy. Although we do not observe a clear, steady average improvement in accuracy when m grows, the results of the 5 corresponding experiments get more concentrated around the average value, giving a desirable reduced variance between experiments. On the other hand, accuracy results at low m show high variance between experiments, which might be accentuated by the fact that nodes features are ignored. However, using node features is without doubt necessary to reach state-of-the-art results, and it is an open question how to incorporate that in our algorithm, but our goal here is mainly to test our algorithm on real data as a proof of concept.

5. CONCLUSION

We proposed a family of algorithms that combines graphlet sampling with efficient mappings. Then, we proposed to choose this mapping as kernel random maps, and showed a concentration of the random embedding around the MMD metric. Finally, while classical random features still require expensive matrix-vector multiplication, we used optical random features projections, which can be computed in $\mathcal{O}(1)$ to get the algorithm's fastest version. Our Experiments showed that it is significantly faster than traditional graphlet kernel and generally performs better while concentrating around the

MMD metric. In our settings, it even outperformed a particular graph convolutional network on graph classification.

A major point left open to be analyzed is how to use our algorithm to classify graphs with node features. One promising possibility is to use our algorithm to generate features embeddings on the graph level, and then feed these embeddings with the nodes' features to a deep neural network. Doing this, we take advantage of the speed of both our algorithm and GPUs. On the theoretical side, the properties of the MMD metric could be further analyzed on particular models of graphs to get a concentration with higher certainty.

6. REFERENCES

- [1] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis, "Matching node embeddings for graph similarity," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [2] Pinar Yanardag and SVN Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1365–1374.
- [3] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.
- [4] Chi Thang Duong, Thanh Dat Hoang, Ha The Hien Dang, Quoc Viet Hung Nguyen, and Karl Aberer, "On node features for graph neural networks," *arXiv preprint arXiv:1911.08795*, 2019.
- [5] X Yan and J Han, "gspan: Graph-based substructure pattern mining, 2002," *Published by the IEEE Computer Society*, 2003.
- [6] Nils M Kriege, Fredrik D Johansson, and Christopher Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, pp. 1–42, 2020.
- [7] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, 2009, pp. 488–495.
- [8] Aman Sinha and John C Duchi, "Learning kernels with random features," in *Advances in Neural Information Processing Systems*, 2016, pp. 1298–1306.
- [9] Johannes Kobler, Uwe Schöning, and Jacobo Torán, *The graph isomorphism problem: its structural complexity*, Springer Science & Business Media, 2012.
- [10] Anna Lubiw, "Some np-complete problems similar to graph isomorphism," *SIAM Journal on Computing*, vol. 10, no. 1, pp. 11–21, 1981.

- [11] Jure Leskovec and Christos Faloutsos, “Sampling from large graphs,” in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 631–636.
- [12] Ali Rahimi and Benjamin Recht, “Random features for large-scale kernel machines,” in *Advances in neural information processing systems*, 2008, pp. 1177–1184.
- [13] Dougal J Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton, “Generative models and model criticism via optimized maximum mean discrepancy,” *arXiv preprint arXiv:1611.04488*, 2016.
- [14] Alaa Saade, Francesco Caltagirone, Igor Carron, Laurent Daudet, Angélique Drémeau, Sylvain Gigan, and Florent Krzakala, “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, 2016, pp. 6215–6219.
- [15] Thorben Funke and Till Becker, “Stochastic block models: A comparison of variants and inference methods,” *PloS one*, vol. 14, no. 4, 2019.
- [16] Paul D Dobson and Andrew J Doig, “Distinguishing enzyme structures from non-enzymes without alignments,” *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.

Appendices

A. PROOF OF THEOREM 1

Proof. We decompose the proof in two steps.

Step 1: infinite s , finite m . First we define the random variables $x_j = |\mathbb{E}_{F \sim S_k(\mathcal{G})} \xi_{w_j}(F) - \mathbb{E}_{F' \sim S_k(\mathcal{G}')} \xi_{w_j}(F')|^2$, which are: i/independent, ii/have expectation $MMD(\mathcal{G}, \mathcal{G}')^2$, /iii are bounded by the interval $[0, 4]$ based on our assumption $|\xi_w| \leq 1$. Thus, as a straight result of applying Hoeffding's inequality with easy manipulation: with probability $1 - \delta$

$$\left| \frac{1}{m} \sum_{j=1}^m x_j - MMD(\mathcal{G}, \mathcal{G}')^2 \right| \leq \frac{4\sqrt{\log(2/\delta)}}{\sqrt{m}} \quad (7)$$

Step 2: finite s and m . For any *fixed* set of random features $\{w_j\}_{1, \dots, m}$ and based on our previous assumptions we have: i/ φ_{RF} is in a ball of radius $M = \frac{\sqrt{m}}{\sqrt{m}} = 1$, ii/ $\mathbb{E}_{F \sim S_k(\mathcal{G})} \varphi(F) = \mathbb{E} \left(\frac{1}{s} \sum_i \varphi(F_i) \right)$. Therefore, we can directly apply the vector version of Hoeffding's inequality on the vectors $\frac{1}{s} \sum_i \varphi(F_i)$ to get that with probability $1 - \delta$:

$$\left\| \mathbb{E}_{F \sim S_k(\mathcal{G})} \varphi(F) - \frac{1}{s} \sum_i \varphi(F_i) \right\| \leq \frac{1 + \sqrt{2 \log \frac{1}{\delta}}}{\sqrt{s}} \quad (8)$$

Defining $J_{exp}(\mathcal{G}, \mathcal{G}') = \|\mathbb{E}_{F \sim S_k(\mathcal{G})} \varphi(F) - \mathbb{E}_{F' \sim S_k(\mathcal{G}')} \varphi(F')\|$ and $J_{avg}(\mathcal{G}, \mathcal{G}') = \|\frac{1}{s} \sum_i \varphi(F_i) - \frac{1}{s} \sum_i \varphi(F'_i)\|$, then using triangular inequality followed by a union bound based on (8), we have the following with probability $1 - 2\delta$,

$$|J_{exp}(\mathcal{G}, \mathcal{G}') - J_{avg}(\mathcal{G}, \mathcal{G}')| \leq \frac{2}{\sqrt{s}} \left(1 + \sqrt{2 \log \frac{1}{\delta}} \right)$$

On the other hand, $J_{exp}(\mathcal{G}, \mathcal{G}') + J_{avg}(\mathcal{G}, \mathcal{G}') \leq 4$, so with same probability:

$$|J_{exp}(\mathcal{G}, \mathcal{G}')^2 - J_{avg}(\mathcal{G}, \mathcal{G}')^2| \leq \frac{8}{\sqrt{s}} \left(1 + \sqrt{2 \log \frac{1}{\delta}} \right) \quad (9)$$

Since it is valid for any fixed set of random features, it is also valid with *joint* probability on random features and samples, by the law of total probability.

Finally, combining (7), (9) with a union bound and a triangular inequality, we have with probability $1 - 3\delta$,

$$\left| \|\varphi(\mathfrak{F}_{\mathcal{G}}) - \varphi(\mathfrak{F}_{\mathcal{G}'})\|^2 - MMD(\mathcal{G}, \mathcal{G}')^2 \right| \leq \frac{4\sqrt{\log(2/\delta)}}{\sqrt{m}} + \frac{8}{\sqrt{s}} \left(1 + \sqrt{2 \log \frac{1}{\delta}} \right)$$

which concludes the proof by taking δ as $\delta/3$. \square