# The Multiscale Laplacian Graph Kernel

**Risi Kondor**
Department of Computer Science
Department of Statistics
University of Chicago
Chicago, IL 60637
risi@cs.uchicago.edu

**Horace Pan**
Department of Computer Science
University of Chicago
Chicago, IL 60637
hopan@uchicago.edu

## Abstract

Many real world graphs, such as the graphs of molecules, exhibit structure at multiple different scales, but most existing kernels between graphs are either purely local or purely global in character. In contrast, by building a hierarchy of nested subgraphs, the Multiscale Laplacian Graph kernels (MLG kernels) that we define in this paper can account for structure at a range of different scales. At the heart of the MLG construction is another new graph kernel, called the Feature Space Laplacian Graph kernel (FLG kernel), which has the property that it can lift a base kernel defined on the vertices of two graphs to a kernel between the graphs. The MLG kernel applies such FLG kernels to subgraphs recursively. To make the MLG kernel computationally feasible, we also introduce a randomized projection procedure, similar to the Nyström method, but for RKHS operators.

## 1 Introduction

There is a wide range of problems in applied machine learning from web data mining [1] to protein function prediction [2] where the input space is a space of graphs. A particularly important application domain is chemoinformatics, where the graphs capture the structure of molecules. In the pharmaceutical industry, for example, machine learning algorithms are regularly used to screen candidate drug compounds for safety and efficacy against specific diseases [3].

Because kernel methods neatly separate the issue of data representation from the statistical learning component, it is natural to formulate graph learning problems in the kernel paradigm. Starting with [4], a number of different graph kernels have appeared in the literature (for an overview, see [5]). In general, a graph kernel $k(\mathcal{G}_1, \mathcal{G}_2)$ must satisfy the following requirements:

1. The kernel should capture the right notion of similarity between $\mathcal{G}_1$ and $\mathcal{G}_2$. For example, if $\mathcal{G}_1$ and $\mathcal{G}_2$ are social networks, then $k$ might capture to what extent their clustering structure, degree distribution, etc. match. If, on the other hand, $\mathcal{G}_1$ and $\mathcal{G}_2$ are molecules, then we are probably more interested in what functional groups are present, and how they are arranged relative to each other.
2. The kernel is usually computed from the adjacency matrices $A_1$ and $A_2$ of the two graphs, but it must be invariant to the ordering of the vertices. In other words, writing the kernel explicitly in terms of $A_1$ and $A_2$, we must have $k(A_1, A_2) = k(A_1, PA_2P^\top)$ for any permutation matrix $P$.

Permutation invariance has proved to be the central constraint around which much of the graph kernels literature is organized, effectively stipulating that graph kernels must be built out of graph invariants. Efficiently computable graph invariants offered by the mathematics literature tend to fall in one of two categories:

1. Local invariants, which can often be reduced to simply counting some local properties, such as the number of triangles, squares, etc. that appear in $\mathcal{G}$ as subgraphs.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2. Spectral invariants, which can be expressed as functions of the eigenvalues of the adjacency matrix or the graph Laplacian.

Correspondingly, while different graph kernels are motivated in very different ways from random walks [4] through shortest paths [6, 7] to Fourier transforms on the symmetric group [8], most graph kernels in the literature ultimately reduce to computing a function of the two graphs that is either purely local or purely spectral. Any of the kernels based on the "subgraph counting" idea (e.g., [9]) are local. On the other hand, most of the random walk based kernels are reducible to a spectral form involving the eigenvalues of either the two graphs individually, or their Kronecker product [5] and therefore are really only sensitive to the large scale structure of graphs.

In practice, it would be desirable to have a kernel that can take structure into account at *multiple different scales*. A kernel between molecules, for example, should not only be sensitive to the overall large-scale shape of the graphs (whether they are more like a chain, a ring, a chain that branches, etc.), but also to what smaller structures (e.g., functional groups) are present in the graphs, and how they are related to the global structure (e.g., whether a particular functional group is towards the middle or one of the ends of a chain).

For the most part, such a multiscale graph kernel has been missing from the literature. Two notable exceptions are the Weisfeiler–Lehman kernel [10] and Propagation Kernel [11]. The WL kernel uses a combination of message passing and hashing to build summaries of the local neighborhoods of vertices at different scales. While shown to be effective, the Weisfeiler–Lehman kernel's hashing step is somewhat ad-hoc; perturbing the edges by a small amount leads to completely different hash features. Similarly, the propagation kernel monitors how the distribution of node/edge labels spreads through the graph and then uses locality sensitivity hashing to efficiently bin the label distributions into feature vectors.

Most recently, structure2vec[12] attempts to represent each graph with a latent variable model and then embeds them into a feature space, using the inner product as a kernel. This approach compares favorably to the standard kernel methods in both accuracy and computational efficiency.

In this paper we present a new graph kernel, the Multiscale Laplacian Graph Kernel (MLG kernel), which, we believe, is the first kernel in the literature that can truly compare structure in graphs simultaneously at multiple different scales. We begin by introducing the Feature Space Laplacian Graph Kernel(FLG kernel) in Section 2. The FLG kernel operates at a single scale, while combining information from the nodes's vertex features with topological information through its Laplacian. An important property of the FLG kernel is that it can work with vertex labels provided as a "base kernel" on the vertices, which allows us to apply the FLG kernel recursively.

The MLG kernel, defined in Section 3, uses the FLG kernel's recursive property to build a hierarchy of subgraph kernels that are sensitive to both the topological relationships between individual vertices, and between subgraphs of increasing sizes. Each kernel is defined in terms of the preceding kernel in the hierarchy. Efficient computability is a major concern in our paper, and recursively defined kernels on combinatorial data structures, can be very expensive. Therefore, in Section 4 we describe a strategy based on a combination of linearizing each level of the kernel (relative to a given dataset) and a randomized low rank projection step, which reduces every stage of the kernel computation to simple operations involving small matrices, leading to a very fast algorithm. Finally, section 5 presents experimental comparisons of our kernel with competing methods.

## 2 Laplacian Graph Kernels

Let $\mathcal{G}$ be a weighted undirected graph with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E$. Recall that the graph Laplacian of $\mathcal{G}$ is an $n \times n$ matrix $L^{\mathcal{G}}$, with

$$L^{\mathcal{G}}_{i,j} = \begin{cases} -w_{i,j} & \text{if } \{v_i, v_j\} \in E \\ \sum_{j\,:\,\{v_i,v_j\}\in E} w_{i,j} & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases}$$

where $w_{i,j}$ is the weight of edge $\{v_i, v_j\}$. The graph Laplacian is positive semi-definite, and in terms of the adjacency matrix $A$ and the weighted degree matrix $D$ it can be expressed as $L = D - A$.

Spectral graph theory tells us that the low eigenvalue eigenvectors of $L^{\mathcal{G}}$ are informative about the overall shape of $\mathcal{G}$. One way of seeing this is to note that for any vector $\mathbf{z} \in \mathbb{R}^n$

$$\mathbf{z}^\top L^{\mathcal{G}} \mathbf{z} = \sum_{\{i,j\} \in E} w_{i,j} (z_i - z_j)^2,$$

so the low eigenvalue eigenvectors are the smoothest functions on $\mathcal{G}$, in the sense that they vary the least between adjacent vertices. An alternative interpretation emerges if we use $\mathcal{G}$ to construct a Gaussian graphical model (Markov Random Field or MRF) over $n$ variables $x_1, \ldots, x_n$ with clique potentials $\phi(x_i, x_j) = e^{-w_{i,j}(x_i - x_j)^2/2}$ for each edge and $\psi(x_i) = e^{-\eta x_i^2/2}$ for each vertex. The joint distribution of $\mathbf{x} = (x_1, \ldots, x_n)^\top$ is then

$$p(\mathbf{x}) \propto \Big( \prod_{\{v_i, v_j\} \in E} e^{-w_{i,j}(x_i - x_j)^2/2} \Big) \Big( \prod_{v_i \in V} e^{-\eta x_i^2/2} \Big) = e^{-\mathbf{x}^\top (L + \eta I)\, \mathbf{x}/2}, \tag{1}$$

showing that the covariance matrix of $\mathbf{x}$ is $(L^{\mathcal{G}} + \eta I)^{-1}$. Note that the $\psi$ factors were added to ensure that the distribution is normalizable, and $\eta$ is typically just a small constant "regularizer": $L^{\mathcal{G}}$ actually has a zero eigenvalue eigenvector (namely the constant vector $n^{-1/2}(1, 1, \ldots, 1)^\top$), so without adding $\eta I$ we would not be able to invert it. In the following we will call $L^{\mathcal{G}} + \eta I$ the regularized Laplacian, and denote it simply by $L$.

Both the above views suggest that if we want define a kernel between graphs that is sensitive to their overall shape, comparing the low eigenvalue eigenvectors of their Laplacians is a good place to start. Following the MRF route, given two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ of $n$ vertices, we can define the kernel between them to be a kernel between the corresponding distributions $p_1 = \mathcal{N}(0, L_1^{-1})$ and $p_2 = \mathcal{N}(0, L_2^{-1})$. Specifically, we will use the Bhattacharyya kernel [13]

$$k(p_1, p_2) = \int \sqrt{p_1(x)} \sqrt{p_2(x)}\, dx, \tag{2}$$

because for Gaussian distributions it can be computed in closed form, giving

$$k(p_1, p_2) = \frac{\left| \left( \frac{1}{2} L_1 + \frac{1}{2} L_2 \right)^{-1} \right|^{1/2}}{\left| L_1^{-1} \right|^{1/4} \left| L_2^{-1} \right|^{1/4}}\ .$$

If some of the eigenvalues of $L_1^{-1}$ or $L_2^{-1}$ are zero or very close to zero, along certain directions in space the two distributions in (2) become very flat, leading to vanishingly small kernel values (unless the "flat" directions of the two Gaussians are perfectly aligned). To remedy this problem, similarly to [14], we "soften" (or regularize) the kernel by adding some small constant $\gamma$ times the identity to $L_1^{-1}$ and $L_2^{-1}$. This leads to what we call the Laplacian Graph Kernel.

**Definition 1.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs with $n$ vertices with (regularized) Laplacians $L_1$ and $L_2$, respectively. We define the* ***Laplacian graph kernel (LG kernel)*** *with parameter $\gamma$ between $\mathcal{G}_1$ and $\mathcal{G}_2$ as*

$$k_{LG}(\mathcal{G}_1, \mathcal{G}_2) = \frac{\left| \left( \frac{1}{2} S_1^{-1} + \frac{1}{2} S_2^{-1} \right)^{-1} \right|^{1/2}}{|S_1|^{1/4} \, |S_2|^{1/4}}\ , \tag{3}$$

*where $S_1 = L_1^{-1} + \gamma I$ and $S_2 = L_2^{-1} + \gamma I$.*

By virtue of (2), the LG kernel is positive semi-definite, and because the value of the overlap integral is largely determined by the extent to which the subspaces spanned by the largest eigenvalue eigenvectors of $L_1^{-1}$ and $L_2^{-1}$ are aligned, it effectively captures similarity between the overall shapes of $\mathcal{G}_1$ and $\mathcal{G}_2$. However, the LG kernel does suffer from three major limitations: it assumes that both graphs have the same number of vertices, it is only sensitive to the overall structure of the two graphs, and it is not invariant to permuting the vertices. Our goal for the rest of this paper is to overcome each of these limitations, while retaining the LG kernel's attractive spectral interpretation.

### 2.1 The feature space Laplacian graph kernel (FLG kernel)

In the probabilistic view of the LG kernel, every graph generates random vectors $\mathbf{x} = (x_1, \ldots, x_n)^\top$ according to (1), and the kernel between two graphs is determined by comparing the corresponding

distributions. The invariance problem arises because the ordering of the variables $x_1, \ldots, x_n$ is arbitrary: even if $\mathcal{G}_1$ and $\mathcal{G}_2$ are topologically the same, $k_{\text{LG}}(\mathcal{G}_1, \mathcal{G}_2)$ might be low if their vertices happen to be numbered differently.

One of the central ideas of this paper is to address this issue by transforming from the "vertex space variables" $x_1, \ldots, x_n$ to "feature space variables" $y_1, \ldots, y_m$, where $y_i = \sum_j t_{i,j}(x_j)$, and each $t_{i,j}$ function may only depend on $j$ through local and reordering invariant properties of vertex $v_j$. If we then compute an analogous kernel to the LG kernel, but now between the distributions of the $\mathbf{y}$'s rather than the $\mathbf{x}$'s, the resulting kernel will be permutation invariant.

In the simplest case, the $t_{i,j}$ functions are linear, i.e., $t_{i,j}(x_j) = \phi_i(v_j) \cdot x_j$, where $(\phi_1, \ldots, \phi_m)$ is a collection of $m$ local (and permutation invariant) vertex features. For example, $\phi_i(v_j)$ may be the degree of vertex $v_j$, or the value of $h^\beta(v_j, v_j)$, where $h$ is the diffusion kernel on $\mathcal{G}$ with length scale parameter $\beta$ (c.f., [15]). In the chemoinformatics setting, the $\phi_i$'s might be some way of encoding what type of atom is located at vertex $v_j$.

The linear transform of a multivariate normal random variable is multivariate normal. In our case, defining $U_{i,j} = \phi_i(v_j)_{i,j}$ and $\mathbf{y} = U\mathbf{x}$, we have $\mathbb{E}(\mathbf{y}) = 0$ and $\text{Cov}(\mathbf{y}, \mathbf{y}) = U\,\text{Cov}(\mathbf{x}, \mathbf{x})U^\top = UL^{-1}U^\top$, leading to the following kernel, which is the workhorse of the present paper.

**Definition 2.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs with regularized Laplacians $L_1$ and $L_2$, respectively, $\gamma \geq 0$ a parameter, and $(\phi_1, \ldots, \phi_m)$ a collection of $m$ local vertex features. Define the corresponding feature mapping matrices*

$$[U_1]_{i,j} = \phi_i(v_j) \qquad [U_2]_{i,j} = \phi_i(v'_j),$$

*where $v_j$ is the $j$'th vertex of $\mathcal{G}_1$ and $v'_j$ is the $j$'th vertex of $\mathcal{G}_2$. The corresponding **Feature space Laplacian graph kernel (FLG kernel)** is defined*

$$k_{FLG}(\mathcal{G}_1, \mathcal{G}_2) = \frac{\left|\left(\frac{1}{2}S_1^{-1} + \frac{1}{2}S_2^{-1}\right)^{-1}\right|^{1/2}}{|S_1|^{1/4}\,|S_2|^{1/4}}\,, \tag{4}$$

*where $S_1 = U_1 L_1^{-1} U_1^\top + \gamma I$ and $S_2 = U_2 L_2^{-1} U_2^\top + \gamma I$.*

Since the $\phi_1, \ldots, \phi_m$ vertex features, by definition, are local and invariant to vertex renumbering, the FLG kernel is permutation invariant. Moreover, the distributions now live in the space of features rather than the space defined by the vertices, so we can apply the kernel to two graphs with different numbers of vertices.

Similarly to the LG kernel, the FLG kernel also captures information about the global shape of graphs. However, whereas, intuitively, the former encodes information such as "$\mathcal{G}$ is an elongated graph with vertex number $i$ towards one and and vertex number $j$ at the other", the FLG kernel can capture information more like "$\mathcal{G}$ is elongated with low degree vertices at one end and high degree vertices at the other". The major remaining shortcoming of the FLG kernel is that it cannot take into account structure at multiple different scales.

### 2.2 The "kernelized" FLG kernel

The key to boosting $k_{\text{FLG}}$ to a multiscale kernel is that it itself can be "kernelized", i.e., it can be computed from just the inner products between the feature vectors of the vertices (which we call the base kernel) without having to know the actual $\phi_i(v_j)$ features values.

**Definition 3.** *Given a collection $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)^\top$ of local vertex features, we define the corresponding **base kernel** $\kappa$ between two vertices $v$ and $v'$ as the dot product of their feature vectors: $\kappa(v, v') = \boldsymbol{\phi}(v) \cdot \boldsymbol{\phi}(v')$.*

Note that in this definition $v$ and $v'$ may be two vertices of the same graph, or of two different graphs. We first show that, similarly to other kernel methods [16], to compute $k_{\text{FLG}}(\mathcal{G}_1, \mathcal{G}_2)$ one only needs to consider the subspace of $\mathbb{R}^m$ spanned by the feature vectors of their vertices.

**Proposition 1.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs with vertex sets $V_1 = \{v_1 \ldots v_{n_1}\}$ and $V_2 = \{v'_1 \ldots v'_{n_2}\}$, and let $\{\xi_1, \ldots, \xi_p\}$ be an orthonormal basis for the subspace*

$$W = \text{span}\{\boldsymbol{\phi}(v_1), \ldots, \boldsymbol{\phi}(v_{n_1}), \boldsymbol{\phi}(v'_1), \ldots, \boldsymbol{\phi}(v'_{n_2})\}.$$

*Then, (4) can be rewritten as*

$$k_{FLG}(\mathcal{G}_1, \mathcal{G}_2) = \frac{\left|\left(\frac{1}{2}\overline{S}_1^{-1} + \frac{1}{2}\overline{S}_2^{-1}\right)^{-1}\right|^{1/2}}{|\overline{S}_1|^{1/4}\,|\overline{S}_2|^{1/4}}\,, \tag{5}$$

*where $[\overline{S}_1]_{i,j} = \xi_i^\top S_1 \xi_j$ and $[\overline{S}_2]_{i,j} = \xi_i^\top S_2 \xi_j$. In other words, $\overline{S}_1$ and $\overline{S}_2$ are the projections of $S_1$ and $S_2$ to $W$.*

Similarly to kernel PCA [17] or the Bhattacharyya kernel [14], the easiest way to construct the basis $\{\xi_1, \ldots, \xi_p\}$ required by (5) is to compute the eigendecomposition of the joint Gram matrix of the vertices of the two graphs.

**Proposition 2.** *Let $\mathcal{G}_1$ and $\mathcal{G}$ be as in Proposition 1, $\overline{V} = \{\overline{v}_1, \ldots, \overline{v}_{n_1+n_2}\}$ be the union of their vertex sets (where it is assumed that the first $n_1$ vertices are $\{v_1, \ldots, v_{n_1}\}$ and the second $n_2$ vertices are $\{v'_1, \ldots, v'_{n_2}\}$), and define the joint Gram matrix $K \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ as*

$$K_{i,j} = \kappa(\overline{v}_i, \overline{v}_j) = \boldsymbol{\phi}(\overline{v}_i)^\top \boldsymbol{\phi}(\overline{v}_j).$$

*Let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$ be (a maximal orthonormal set of) the non-zero eigenvalue eigenvectors of $K$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_p$. Then the vectors*

$$\xi_i = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{n_1+n_2} [\boldsymbol{u}_i]_\ell \, \boldsymbol{\phi}(\overline{v}_\ell) \tag{6}$$

*form an orthonormal basis for $W$. Moreover, defining $Q = [\lambda_1^{1/2}\boldsymbol{u}_1, \ldots, \lambda_p^{1/2}\boldsymbol{u}_p] \in \mathbb{R}^{p\times p}$ and setting $Q_1 = Q_{1:n_1,:}$ and $Q_2 = Q_{n_1+1:n_2,:}$ (the first $n_1$ and remaining $n_2$ rows of $Q$, respectively), the matrices $\overline{S}_1$ and $\overline{S}_2$ appearing in (5) can be computed as*

$$\overline{S}_1 = Q_1^\top L_1^{-1} Q_1 + \gamma I, \qquad \overline{S}_2 = Q_2^\top L_2^{-1} Q_2 + \gamma I. \tag{7}$$

Proofs of these two propositions are given in the Supplemental Material. As in other kernel methods, the significance of Propositions 1 and 2 is not just that they show how $k_{\text{FLG}}(\mathcal{G}_1, \mathcal{G}_2)$ can be efficiently computed when $\boldsymbol{\phi}$ is very high dimensional, but that they make it clear that the FLG kernel may be induced from any base kernel. For completeness, we close this section with this generalized definition of the FLG kernel.

**Definition 4.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs. Assume that each of their vertices comes from an abstract vertex space $\mathcal{V}$ and that $\kappa\colon \mathcal{V}\times\mathcal{V} \to \mathbb{R}$ is a symmetric positive semi-definite kernel on $\mathcal{V}$. The **generalized FLG kernel** induced from $\kappa$ is then defined as*

$$k^\kappa_{FLG}(\mathcal{G}_1, \mathcal{G}_2) = \frac{\left|\left(\frac{1}{2}\overline{S}_1^{-1} + \frac{1}{2}\overline{S}_2^{-1}\right)^{-1}\right|^{1/2}}{|\overline{S}_1|^{1/4}\,|\overline{S}_2|^{1/4}}\,, \tag{8}$$

*where $\overline{S}_1$ and $\overline{S}_2$ are as defined in Proposition 2.*

## 3 The multiscale Laplacian graph kernel (MLG kernel)

By multiscale graph kernel we mean a kernel that is able to capture similarity between graphs not just based on the topological relationships between their individual vertices, but also the topological relationships between subgraphs. The key property of the FLG kernel that allows us to build such a kernel is that it can be applied recursively. In broad terms, the construction goes as follows:

1. Given a graph $\mathcal{G}$, divide it into a large number of small (typically overlapping) subgraphs and compute the FLG kernel between any two subgraphs.
2. Each subgraph is attached to some vertex of $\mathcal{G}$ (for example, its center), so we can reinterpret the FLG kernel as a new base kernel between the vertices.
3. Now divide $\mathcal{G}$ into larger subgraphs, compute the new FLG kernel between them induced from the new base kernel, and recurse.

To compute the actual multiscale graph kernel $\mathfrak{K}$ between $\mathcal{G}$ and another graph $\mathcal{G}'$, we follow the same process for $\mathcal{G}'$ and then set $\mathfrak{K}(\mathcal{G}, \mathcal{G}')$ equal to the FLG kernel induced from their top level base kernels. The following definitions formalize this construction.

**Definition 5.** *Let $\mathcal{G}$ be a graph with vertex set $V$, and $\kappa$ a positive semi-definite kernel on $V$. Assume that for each $v \in V$ we have a nested sequence of $L$ neighborhoods*

$$v \in N_1(v) \subseteq N_2(v) \subseteq \ldots \subseteq N_L(v) \subseteq V, \tag{9}$$

*and for each $N_\ell(v)$, let $G_\ell(v)$ be the corresponding induced subgraph of $\mathcal{G}$. We define the **Multiscale Laplacian Subgraph Kernels (MLS kernels)**, $\mathfrak{K}_1, \ldots, \mathfrak{K}_L\colon V\times V \to \mathbb{R}$ as follows:*

1. $\mathfrak{K}_1$ *is just the FLG kernel* $k^{\kappa}_{FLG}$ *induced from the base kernel* $\kappa$ *between the lowest level subgraphs:*

$$\mathfrak{K}_1(v, v') = k^{\kappa}_{FLG}(G_1(v), G_1(v')).$$

2. *For* $\ell = 2, 3, \ldots, L$, $\mathfrak{K}_\ell$ *is the FLG kernel induced from* $\mathfrak{K}_{\ell-1}$ *between* $G_\ell(v)$ *and* $G_\ell(v')$*:*

$$\mathfrak{K}_\ell(v, v') = k^{\mathfrak{K}_{\ell-1}}_{FLG}(G_\ell(v), G_\ell(v')).$$

Definition 5 defines the MLS kernel as a kernel between different subgraphs of the same graph $\mathcal{G}$. However, if two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ share the same base kernel, the MLS kernel can also be used to compare any subgraph of $\mathcal{G}_1$ with any subgraph of $\mathcal{G}_2$. This is what allows us to define an $L+1$'th FLG kernel, which compares the two full graphs.

**Definition 6.** *Let* $\mathfrak{G}$ *be a collection of graphs such that all their vertices are members of an abstract vertex space* $\mathcal{V}$ *endowed with a symmetric positive semi-definite kernel* $\kappa\colon \mathcal{V} \times \mathcal{V} \to \mathbb{R}$*. Assume that the MLS kernels* $\mathfrak{K}_1, \ldots, \mathfrak{K}_L$ *are defined as in Definition 5, both for pairs of subgraphs within the same graph and across pairs of different graphs. We define the* ***Multiscale Laplacian Graph Kernel (MLG kernel)*** *between any two graphs* $\mathcal{G}_1, \mathcal{G}_2 \in \mathfrak{G}$ *as*

$$\mathfrak{K}(\mathcal{G}_1, \mathcal{G}_2) = k^{\mathfrak{K}_L}_{FLG}(\mathcal{G}_1, \mathcal{G}_2).$$

Definition 6 leaves open the question of how the neighborhoods $N_1(v), \ldots, N_L(v)$ are to be defined. In the simplest case, we set $N_\ell(v)$ to be the ball $B_r(v)$ (i.e., the set of vertices at a distance at most $r$ from $v$), where $r = r_0 d^{\ell-1}$ for some $d > 1$.

### 3.1 Computational complexity

Definitions 5 and 6 suggest a recursive approach to computing the MLG kernel: computing $\mathfrak{K}(\mathcal{G}_1, \mathcal{G}_2)$ first requires computing $\mathfrak{K}_L(v, v')$ between all $\binom{n_1+n_2}{2}$ pairs of top level subgraphs across $\mathcal{G}_1$ and $\mathcal{G}_2$; each of these kernel evaluations requires computing $\mathfrak{K}_{L-1}(v, v')$ between up to $O(n^2)$ level $L-1$ subgraphs, and so on. Following this recursion blindly would require up to $O(n^{2L+2})$ kernel evaluations, which is clearly infeasible.

The recursive strategy is wasteful because it involves evaluating the same kernel entries over and over again in different parts of the recursion tree. An alternative solution that requires only $O(Ln^2)$ kernel evaluations would be to first compute $\mathfrak{K}_1(v, v')$ for *all* $(v, v')$ pairs, then compute $\mathfrak{K}_2(v, v')$ for *all* $(v, v')$ pairs and so on.

## 4 Linearized Kernels and Low Rank Approximation

Computing the MLG kernel between two graphs, as described in the previous section, may involve $O(Ln^2)$ kernel evaluations. At the top levels of the hierarchy each $G_\ell(v)$ might have $\Theta(n)$ vertices, so the cost of a single FLG kernel evaluation can be as high as $O(n^3)$. Somewhat pessimistically, this means that the overall cost of computing $k_{\text{FLG}}(\mathcal{G}_1, \mathcal{G}_2)$ is $O(Ln^5)$. Given a dataset of $M$ graphs, computing their Gram matrix requires repeating this for all $\{\mathcal{G}_1, \mathcal{G}_2\}$ pairs, giving $O(LM^2n^5)$, which is even more problematic. The solution that we propose in this section is to compute for each level $\ell = 1, 2, \ldots, L+1$ a single joint basis for *all* subgraphs at the given level across *all* graphs $\mathcal{G}_1, \ldots, \mathcal{G}_M$. For concreteness, we go back to the definition of the FLG kernel.

**Definition 7.** *Let* $\mathfrak{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ *be a collection of graphs,* $V_1, \ldots, V_M$ *their vertex sets, and assume that* $V_1, \ldots, V_M \subseteq \mathcal{V}$ *for some general vertex space* $\mathcal{V}$*. Further, assume that* $\kappa\colon \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ *is a positive semi-definite kernel on* $\mathcal{V}$*,* $\mathcal{H}_\kappa$ *is its Reproducing Kernel Hilbert Space, and* $\phi\colon \mathcal{V} \to \mathcal{H}_\kappa$ *is the corresponding feature map satisfying* $\kappa(v, v') = \langle \phi(v), \phi(v') \rangle$ *for any* $v, v' \in \mathcal{V}$*. The* ***joint vertex feature space*** *of* $\{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ *is then* $W_{\mathfrak{G}} = \operatorname{span}\big\{ \bigcup_{i=1}^{M} \bigcup_{v \in V_i} \{\phi(v)\} \big\}$.

$W_{\mathfrak{G}}$ is just the generalization of the $W$ space defined in Proposition 1 from two graphs to $M$. The following generalization of Propositions 1 and 2 is then immediate.

**Proposition 3.** *Let* $N = \sum_{i=1}^{M} |V_i|$, $\overline{V} = (\overline{v}_1, \ldots, \overline{v}_N)$ *be the concatenation of the vertex sets* $V_1, \ldots, V_M$*, and* $K$ *the corresponding joint Gram matrix* $K_{i,j} = \kappa(\overline{v}_i, \overline{v}_j) = \langle \phi(\overline{v}_i), \phi(\overline{v}_j) \rangle$ *. Let*

*$\boldsymbol{u}_1, \ldots, \boldsymbol{u}_P$ be a maximal orthonormal set of non-zero eigenvalue eigenvectors of $K$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_P$. Then the vectors*

$$\xi_i = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{N} [\boldsymbol{u}_i]_\ell \, \phi(\overline{v}_\ell) \qquad i = 1, \ldots, P$$

*form an orthonormal basis for $W_{\mathfrak{G}}$. Moreover, defining $Q = [\lambda_1^{1/2}\boldsymbol{u}_1, \ldots, \lambda_p^{1/2}\boldsymbol{u}_p] \in \mathbb{R}^{P \times P}$, and setting $Q_1$ to be the submatrix of $Q$ composed of its first $|V_1|$ rows; $Q_2$ be the submatrix composed of the next $|V_2|$ rows, and so on, for any $\mathcal{G}_i, \mathcal{G}_j \in \mathfrak{G}$, the generalized FLG kernel induced from $\kappa$ (Definition 4) can be expressed as*

$$k_{FLG}(\mathcal{G}_i, \mathcal{G}_j) = \frac{\left| \left( \frac{1}{2}\overline{S}_i^{-1} + \frac{1}{2}\overline{S}_j^{-1} \right)^{-1} \right|^{1/2}}{|\overline{S}_i|^{1/4} \, |\overline{S}_j|^{1/4}}, \tag{10}$$

*where $\overline{S}_i = Q_i^\top L_i^{-1} Q_i + \gamma I$ and $\overline{S}_j = Q_j^\top L_j^{-1} Q_j + \gamma I$.*

The significance of Proposition 3 is that $S_1, \ldots, S_M$ are now fixed matrices that do not need to be recomputed for each kernel evaluation. Once we have constructed the joint basis $\{\xi_1, \ldots, \xi_P\}$, the $S_i$ matrix of each graph $\mathcal{G}_i$ can be computed independently, as a precomputation step, and individual kernel evaluations reduce to just plugging them into (10). At a conceptual level, Proposition 3 *linearizes* the kernel $\kappa$ by projecting everything down to $W_{\mathfrak{G}}$. In particular, it replaces the $\{\phi(\overline{v}_i)\}$ RKHS vectors with explicit finite dimensional feature vectors given by the corresponding rows of $Q$, just like we had in the "unkernelized" FLG kernel of Definition 2.

For our multiscale kernels this is particularly important, because linearizing not just $k_{\text{FLG}}^{\kappa}$, but also $k_{\text{FLG}}^{\mathfrak{K}_1}, k_{\text{FLG}}^{\mathfrak{K}_2}, \ldots$, allows us to compute the MLG kernel level by level, without recursion. After linearizing the base kernel $\kappa$, we attach explicit, finite dimensional vectors to each vertex of each graph. Then we compute compute $k_{\text{FLG}}^{\mathfrak{K}_1}$ between all pairs of lowest level subgraphs, and linearizing this kernel as well, each vertex effectively just gets an updated feature vector. Then we repeat the process for $k_{\text{FLG}}^{\mathfrak{K}_2} \ldots k_{\text{FLG}}^{\mathfrak{K}_L}$, and finally we compute the MLG kernel $\mathfrak{K}(\mathcal{G}_1, \mathcal{G}_2)$.

### 4.1 Randomized low rank approximation

The difficulty in the above approach of course is that at each level (3) is a Gram matrix between *all* vertices of *all* graphs, so storing it is already very costly, let along computing its eigendecomposition. Morever, $P = \dim(W_{\mathfrak{G}})$ is also very large, so managing the $\overline{S}_1, \ldots, \overline{S}_M$ matrices (each of which is of size $P \times P$) becomes infeasible. The natural alternative is to replace $W_{\mathfrak{G}}$ by a smaller, *approximate* joint features space, defined as follows.

**Definition 8.** *Let $\mathfrak{G}, \kappa, \mathcal{H}_\kappa$ and $\phi$ be defined as in Definition 7. Let $\tilde{V} = (\tilde{v}_1, \ldots, \tilde{v}_{\tilde{N}})$ be $\tilde{N} \ll N$ vertices sampled from the joint vertex set $\overline{V} = (\overline{v}_1, \ldots, \overline{v}_N)$. Then the corresponding **subsampled vertex feature space** is*

$$\tilde{W}_{\mathfrak{G}} = \operatorname{span}\{ \, \phi(\tilde{v}) \mid \tilde{v} \in \tilde{V} \, \}.$$

Similarly to before, we construct an orthonormal basis $\{\xi_1, \ldots, \xi_P\}$ for $\tilde{W}$ by forming the (now much smaller) Gram matrix $\tilde{K}_{i,j} = \kappa(\tilde{v}_i, \tilde{v}_j)$, computing its eigenvalues and eigenvectors, and setting $\xi_i = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{\tilde{N}} [\boldsymbol{u}_i]_\ell \, \phi(\tilde{v}_\ell)$. The resulting approximate FLG kernel is

$$k_{\text{FLG}}(\mathcal{G}_i, \mathcal{G}_j) = \frac{\left| \left( \frac{1}{2}\tilde{S}_i^{-1} + \frac{1}{2}\tilde{S}_j^{-1} \right)^{-1} \right|^{1/2}}{|\tilde{S}_i|^{1/4} \, |\tilde{S}_j|^{1/4}}, \tag{11}$$

where $\tilde{S}_i = \tilde{Q}_i^\top L_i^{-1} \tilde{Q}_i + \gamma I$ and $\tilde{S}_j = \tilde{Q}_j^\top L_j^{-1} \tilde{Q}_j + \gamma I$ are the projections of $\overline{S}_i$ and $\overline{S}_j$ to $\tilde{W}_{\mathfrak{G}}$. We introduce a further layer of approximation by restricting $\tilde{W}_{\mathfrak{G}}$ to be the space spanned by the first $\tilde{P} < P$ basis vectors (ordered by descending eigenvalue), effectively doing kernel PCA on $\{\phi(\tilde{v})\}_{\tilde{v} \in \tilde{V}}$, equivalently, a low rank approximation of $\tilde{K}$.

Assuming that $v_j^g$ is the $j$'th vertex of $\mathcal{G}_g$, in contrast to Proposition 2, now the $j$'th row of $\tilde{Q}_s$ consists of the coordinates of the *projection* of $\phi(v_j^g)$ onto $\tilde{W}_{\mathfrak{G}}$, i.e.,

$$[\tilde{Q}^g]_{j,i} = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{\tilde{N}} [\boldsymbol{u}_i]_\ell \, \left\langle \phi(v_j^g), \phi(\tilde{v}_N) \right\rangle = \frac{1}{\sqrt{\lambda_i}} \sum_{\ell=1}^{\tilde{N}} [\boldsymbol{u}_i]_\ell \, \kappa(v_j^g, \tilde{v}_N).$$

Table 1: Classification Results (Mean Accuracy $\pm$ Standard Deviation)

| Method | MUTAG[21] | PTC[22] | ENZYMES[2] | PROTEINS[2] | NCI1[23] | NCI109[23] |
|---|---|---|---|---|---|---|
| WL | $84.50(\pm 2.16)$ | $59.97(\pm 1.60)$ | $53.75(\pm 1.37)$ | $75.43(\pm 1.95)$ | $\mathbf{84.76(\pm 0.32)}$ | $\mathbf{85.12(\pm 0.29)}$ |
| WL-Edge | $82.94(\pm 2.33)$ | $60.18(\pm 2.19)$ | $52.00(\pm 0.72)$ | $73.63(\pm 2.12)$ | $\mathbf{84.65(\pm 0.25)}$ | $\mathbf{85.32(\pm 0.34)}$ |
| SP | $\mathbf{85.50(\pm 2.50)}$ | $59.53(\pm 1.71)$ | $42.31(\pm 1.37)$ | $75.61(\pm 0.45)$ | $73.61(\pm 0.36)$ | $73.23(\pm 0.26)$ |
| Graphlet | $82.44(\pm 1.29)$ | $55.88(\pm 0.31)$ | $30.95(\pm 0.73)$ | $71.63(\pm 0.33)$ | $62.40(\pm 0.27)$ | $62.35(\pm 0.28)$ |
| $p$–RW | $80.33(\pm 1.35)$ | $59.85(\pm 0.95)$ | $28.17(\pm 0.76)$ | $71.67(\pm 0.78)$ | TIMED OUT | TIMED OUT |
| MLG | $84.21(\pm 2.61)$ | $\mathbf{63.62(\pm 4.69)}$ | $\mathbf{57.92(\pm 5.39)}$ | $\mathbf{76.14(\pm 1.95)}$ | $80.83(\pm 1.29)$ | $81.30(\pm 0.80)$ |

The above procedure is similar to the popular Nyström approximation for kernel matrices [18, 19], except that in our case the ultimate goal is not to approximate the Gram matrix (3), but the $S_1, \ldots, S_M$ matrices used to form the FLG kernel. In practice, we found that the eigenvalues of $K$ usually drop off very rapidly, suggesting that $W$ can be safely approximated by a surprisingly small dimensional subspace ($\tilde{P} \sim 10$), and correspondingly the sample size $\tilde{N}$ can be kept quite small as well (on the order of 100). The combination of these two factors makes computing the entire stack of kernels feasible, reducing the complexity of computing the Gram matrix for a dataset of $M$ graphs of $\theta(n)$ vertices each to $O(ML\tilde{N}^2\tilde{P}^3 + ML\tilde{N}^3 + M^2\tilde{P}^3)$. It is also important to note that this linearization step requires the graphs(not the labels) in the test set to be known during training in order to project the features of the test graphs onto the low rank approximation of $W_{\mathfrak{G}}$.

## 5 Experiments

We tested the efficacy of the MLG kernel by performing classification on benchmark bioinformatics datasets using a binary C-SVM solver [20], and compared our classification results against those from other representative graph kernels from the literature: the Weisfeiler–Lehman Kernel, the Weisfeiler–Lehman Edge Kernel [9], the Shortest Path Kernel [6], the Graphlet Kernel [9], and the $p$-random Walk Kernel [5].

We randomly selected 20% of each dataset to be used as a test set. On the other 80% we did 10 fold cross validation to select the parameters for each kernel method to be used on the test set and repeated this setup 10 times. For the Weisfeiler–Lehman kernels, the height parameter $h$ is chosen from $\{1, 2, ..., 5\}$, the random walk size $p$ for the $p$-random walk kernel was chosen from $\{1, 2, ..., 5\}$, for the Graphlets kernel the graphlet size $n$ was chosen from $\{3, 4, 5\}$. For the parameters of the MLG kernel: we chose $\eta$ from $\{0.01, 0.1, 1\}$, radius size $n$ from $\{1, 2, 3\}$, number of levels $l$ from $\{1, 2, 3\}$, and fixed gamma to be $0.01$. For the MLG kernel, we used the given discrete node labels to create a one-hot binary feature vector for each node and used the dot product between nodes' binary feature vector labels as the base kernel. All experiments were done on a 16 core Intel E5-2670 @ 2.6GHz processor with 32 GB of memory.

We are fairly competitive in accuracy for all datasets except NCI1, and NCI109, where it performs better than all non-Weisfeiler Lehman kernels. The Supplementary Materials give a more detailed discussion of the experiments and datasets.

## 6 Conclusions

In this paper we have proposed two new graph kernels: (1) The FLG kernel, which is a very simple single level kernel that combines information attached to the vertices with the graph Laplacian; (2) The MLG kernel, which is a multilevel, recursively defined kernel that captures topological relationships between not just individual vertices, but also subgraphs. Clearly, designing kernels that can optimally take into account the multiscale structure of actual chemical compounds is a challenging task that will require further work and domain knowledge. However, it is encouraging that even just "straight out of the box", tuning only two or three parameters, the MLG kernel is competitive with other well known kernels in the literature. Beyond just graphs, the general idea of multiscale kernels is of interest for other types of data as well (such as images) that have multiresolution structure, and the way that the MLG kernel chains together local spectral analysis at multiple scales is potentially applicable to these domains as well, which will be the subject of further research.

## References


[1] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

[2] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. In *Proceedings of Intelligent Systems in Molecular Biology (ISMB)*, Detroit, USA, 2005.

[3] H. Kubinyi. Drug research: myths, hype and reality. *Nature Reviews: Drug Discovery*, 2(8):665–668, August 2003.

[4] T. Gärtner. Exponential and geometric kernels for graphs. In *NIPS*02 workshop on unreal data*, volume Principles of modeling nonvectorial data, 2002.

[5] S. V. N. Vishwanathan, Karsten Borgwardt, Risi Kondor, and Nicol Schraudolph. On graph kernels. *Journal of Machine Learning Research (JMLR)*, 11, 2010.

[6] Karsten M. Borgwardt and Hans Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining(ICDM) 2005), 27-30 November 2005, Houston, Texas, USA*, pages 74–81, 2005.

[7] Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten M. Borgwardt. Scalable kernels for graphs with continuous attributes. In *Advances in Neural Information Processing Systemss*, 2013.

[8] Risi Kondor and Karsten Borgwardt. The skew spectrum of graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 496–503. ACM, 2008.

[9] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pages 488–495, 2009.

[10] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research(JMLR)*, 12:2539–2561, November 2011.

[11] Marion Neumann, Roman Garnett, Christian Baukhage, and Kristian Kersting. Propagation kernels: efficient graph kernels from propagated information. In *Machine Learning*, 2016.

[12] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016.

[13] Tony Jebara and Risi Kondor. Bhattacharyya and expected likelihood kernels. In B. Schölkopf and M. Warmuth, editors, *Proceedings of the Annual Conference on Computational Learning Theory and Kernels Workshop (COLT/KW)*, number 2777 in Lecture Notes in Computer Science, pages 57–71, Heidelberg, Germany, 2003. Springer-Verlag.

[14] Risi Kondor and Tony Jebara. A kernel between sets of vectors. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

[15] Marc Alexa, Michael Kazhdan, and Leonidas Guibas. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. In *Processing of Eurographics Symposium on Geometry Processing*, volume 28, 2009.

[16] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

[17] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, Matthias Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 536–542. MIT Press, 1999.

[18] Christopher K. I. Williams and Mattias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[19] Petros Drineas and Michael W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[20] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 3, 2011.

[21] A.K. Debnat, R. L. Lopez de Compadre, G. Debnath, A. j. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J Med Chem*, 34:786–97, 1991.

[22] H.Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma. Statistical evaluation of the predictive toxicology challenge. *Bioinformatics*, pages 1183–1193, 2003.

[23] N. Wale, I. A. Watson, and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, pages 347–375, 2008.