Image Classification with Segmentation Graph Kernels

Zaïd Harchaoui *
LTCI
CNRS & Télécom Paris
46, rue Barrault
75634 Paris cedex 13, France
zaid.harchaoui@enst.fr

Francis Bach
Center for Mathematical Morphology
Ecole Nationale Supérieure des Mines de Paris
35, rue Saint-Honoré
77305 Fontainebleau, France
francis.bach@mines.org

Abstract

We propose a family of kernels between images, defined as kernels between their respective segmentation graphs. The kernels are based on soft matching of subtree-patterns of the respective graphs, leveraging the natural structure of images while remaining robust to the associated segmentation process uncertainty. Indeed, output from morphological segmentation is often represented by a labelled graph, each vertex corresponding to a segmented region, with edges joining neighboring regions. However, such image representations have mostly remained underused for learning tasks, partly because of the observed instability of the segmentation process and the inherent hardness of inexact graph matching with uncertain graphs. Our kernels count common virtual substructures amongst images, which enables to perform efficient supervised classification of natural images with a support vector machine. Moreover, the kernel machinery allows us to take advantage of recent advances in kernel-based learning: i) semi-supervised learning reduces the required number of labelled images, while ii) multiple kernel learning algorithms efficiently select the most relevant similarity measures between images within our family.

1. Introduction

Image classification as a machine learning task enjoys numerous applications, such as image retrieval or object recognition. Images are naturally high-dimensional data, which demands mandatory pre-processing targeted towards dimensionality reduction. Most techniques require a pre-processing step, which can be global as in color histogram binning [3], or local through feature extraction [20]. We are then left with a trade-off between the extreme cases of low-level/high-level or high-level/low-level pre-processing and subsequent analysis.

Convolutional nets and graph-transformer networks managed this trade-off by merging both steps in a single end-to-end process while achieving excellent performances on benchmarks [19]. As emphasized in [13], robust empirical results of non-convex architectures thus far are in contrast with a lack of theoretical analysis on their performances. Kernel-based methods such as support vector machines (SVM) are simpler to analyze and more flexible to use since they rely upon a convex minimization criterion [29]. However, they are rather limited for processing complex structured data such as images when used with standard kernels such as the Gaussian kernel applied on global features. Indeed, most applications of SVMs to image classification rely on the extraction of a large set of handcrafted features and learning on those features from a large or very large dataset of labelled images [26]. Taking into account the inner structure of images in a kernel-based method would allow to significantly reduce the required number of training images, while still benefitting from the theoretical and practical advantages of kernel-based methods. In particular, recent advances in kernel-based learning, such as multiple kernel learning [1] and semi-supervised learning [4], may then be brought to bear in order to simplify the learning procedure, by reducing the number of hyper-parameters and the number of labelled examples.

An emerging line of research, consisting in designing meaningful kernels on structured data, produced promising results in bioinformatics and natural language processing [29]. Namely, graph kernels have recently received an increased attention [9, 16], and showed good performances for classification of small molecules [21]. We attempt here to tackle the issue of image classification with kernel-based methods, by using segmentation graphs obtained by morphological segmentation. If enough segments are used, i.e., if the image (and hence the objects of interest) is *over-*

*Part of this work was done during an intership of the first author at *Centre de Morphologie Mathématique* of *École Nationale Supérieure des Mines de Paris*.

segmented, then the segmentation enables to reduce the dimension of the image while preserving the boundaries of objects. Image dimensionality goes from millions of pixels down to hundreds of segments, with little loss of information. Those segments are naturally embedded in a planar graph structure. Our approach takes into account graph planarity. The main theme of this paper is to feed kernel-based learning methods with kernels measuring appropriate segmentation graph similarity.

In Section 2, we present the segmentation algorithm that we have used and the resulting graph representation of images. In Section 3, we review previous work on graph matching for computer vision as well as on kernels for images, while in Section 4, we present our family of kernels based on string and tree patterns of the graphs, with detailed specifications for images detailed in Section 5. We present experiments on several classification benchmarks in Section 6, with in particular applications of semi-supervised learning and multiple kernel learning.

2. Morphological segmentation graphs

Among the many methods available for the segmentation of natural images [23, 30, 5], we chose to work within the watershed transform framework [23], which allows a fast segmentation of large images into a given number of segments. First, given a color image, a gray-scale gradient image is computed from oriented energy filters on the LAB representation of the image, with two different scales and eight orientations [22]. Then, the watershed transform is applied to this gradient image, and the number of resulting regions is reduced to a given number p ($p = 100$ in our simulations) using the hierarchical framework of [23]. We finally obtain p singly-connected regions, together with the planar neighborhood graph. An Example of segmentation is shown in Figure 1.

We have chosen a large value of p because our similarity measure implicitly relies on the fact that images are mostly over-segmented, i.e., the objects of interest may span more than one segment, but very few segments span several objects. This has to be contrasted with the usual (and often unreached) segmentation goal of obtaining one segment per object. In this paper, we always use the same number of segments; a detailed analysis of the effect of choosing a different number of segments, or a number which depends on the given image, is outside the scope of this paper.

In the remaining of the paper, all images will be represented by a *segmentation graph*, i.e. an undirected labelled planar graph, where each vertex is one singly connected region, with edges joining neighboring regions. Since our graph is obtained from neighboring singly connected regions, the graph is planar, i.e., it can be embedded in a plane where no edge intersects [7]. The only property of planar graphs that we are going to use in this paper is the fact that for each vertex in the graph, there is a natural notion of cyclic ordering of the vertices (see Figure 5). Another typical feature of our graphs is their sparsity: indeed, the degree (number of neighbors) of each node is usually small, and the maximum degree typically does not exceed 10.

There are many ways to assign labels to regions, based on shape, color or texture, leading to highly multivariate labels, which is to be contrasted with the usual application of structured kernels in bioinformatics or natural language processing, where labels belong to a small discrete set. Our family of kernels simply require a simple kernel $k(\ell, \ell')$ between labels ℓ and ℓ' of different regions, which can be any positive semi-definite kernel [29], and not merely a Dirac kernel which is commonly used for exact matching in bioinformatics applications. We could base such a kernel on any relevant visual information; in this paper, as shown in Section 5, we considerred kernels between color histograms of each segment, as well as weights corresponding to the size (area) of the segment. Note that our family of kernels could also take into account kernels between labels defined on edges of the graphs.

3. Previous work

The idea of matching graphs for the purpose of image classification has a long history in computer vision, and has been investigated by many authors [28, 10]. However, the general *graph matching* problem is especially hard as most of the simple operations which are simple on strings (such as matching and edit distances) are NP-hard for general undirected graphs. Namely, while exact graph matching is unrealistic in our context, inexact graph matching is NP-hard and subgraph matching is NP-complete [7]. Hence most work so far has focused on finding ingenious approximate solutions to this challenging problem [14]. An interesting line of research consists in overcoming graph matching problems by projecting graphs on strings by the so-called seriation procedure [28], and then use string edit distance [11] as a proxy to graph edit distance.

Designing kernels for image classification is also an active research topic. The work of [3] investigated image classification with kernels on color histograms. The bag-of-pixels kernels proposed in [15, 6] compare color distribution of two images by using kernels between probability measures, and was extended to hierarchical multi-scale settings in [6, 18]. A different approach, taking into account the non-exchangeability of pixels in an image, is the one of [25] where the edit distance between two graphs is directly plugged into a kernel to defined a similarity measure between graphs. Last, kernels between shock graphs for pedestrian detection designed in [31] were an attempt to capture the topological structure of images.

Our method efficiently circumvents the graph matching

problem by soft-matching tree-walks, i.e. virtual substructures of graphs, in order to obtain kernels computable in polynomial time. Our kernels keep the underlying topological structure of graphs by describing them through tree-walks, while in graph seriation the topological structure somehow fades away by only keeping one particular substring of the graph. Moreover our kernels encompasses local information, by using segment histograms as local features, as well as global information, by summing up all soft matchings of tree-walks.

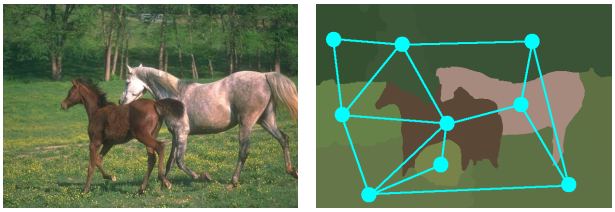

Figure 1. An example of natural image (left), and its segmentation mosaic (right) obtained by using the median RGB color in each segmented region. The associated segmentation graph is depicted in light green

4. Segmentation graph kernels

In this section, we present our family of kernels for planar labelled graphs; we consider two planar labelled graphs, $\mathbf{G}$ and $\mathbf{H}$ with vertex sets $\mathcal{V}_\mathbf{G}$ and $\mathcal{V}_\mathbf{H}$ and edge sets $\mathcal{E}_\mathbf{G}$ and $\mathcal{E}_\mathbf{H}$. We assume that we have labelling functions $\ell_\mathbf{G} : \mathcal{V}_\mathbf{G} \mapsto \mathcal{L}$ and $\ell_\mathbf{H} : \mathcal{V}_\mathbf{H} \mapsto \mathcal{L}$ from the vertex sets to a set of labels $\mathcal{L}$. In this paper, the label set will be a large continuous multivariate set (e.g., histograms of colors and segment areas).

4.1. Paths and walks

Given a graph $\mathbf{G}$, a *walk* is a finite sequence of neighboring vertices, while a *path* is a walk such that all its vertices are distinct. See Figure 2 and Figure 3 for an enumeration of paths and walks for a given undirected graph. Paths and walks are similar notions with different intuitive interpretations and computational properties. On the one hand, paths corresponds to the intuitive notion of substrings of a graph, but most operations involving the set of substrings of a graph leads to NP-hard problems [7]. On the other hand, walks are slightly less intuitive as they can go back and forth along the graph, but they readily lead to efficient polynomial time algorithms, as we now present.

4.2. Walk kernels

Let us denote $\mathcal{W}^p_\mathbf{G}$ (resp. $\mathcal{W}^p_\mathbf{H}$) the set of walks of length p in $\mathbf{G}$ (resp. $\mathbf{H}$). We assume that we are given a *basis kernel* on labels $k(\ell, \ell')$. The p-th order walk kernel $k^p_\mathcal{W}(\mathbf{G}, \mathbf{H})$

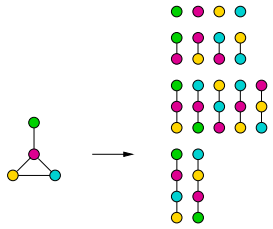

Figure 2. Enumeration of paths from a graph, ordered by lengths. Each color represents a different label.

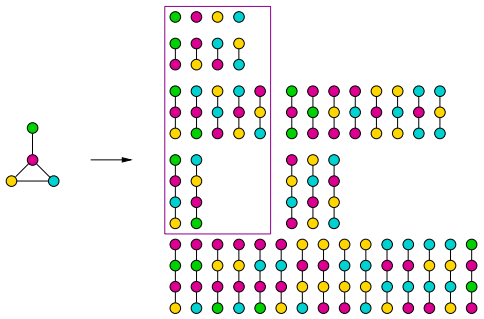

Figure 3. Enumeration of walks from a graph, ordered by lengths. Each color represents a different label. Walks lying within the box are actually paths.

between $\mathbf{G}$ and $\mathbf{H}$ is defined as

$$k^p_\mathcal{W}(\mathbf{G}, \mathbf{H}) = \sum_{\substack{(r_1, \ldots, r_p) \in \mathcal{W}^p_\mathbf{G} \\ (s_1, \ldots, s_p) \in \mathcal{W}^p_\mathbf{H}}} \prod_{i=1}^{p} k(\ell_\mathbf{G}(r_i), \ell_\mathbf{H}(s_i)).$$

When the basis kernel is a Dirac kernel (a situation usually referred to as exact matching), $k^p_\mathcal{W}(\mathbf{G}, \mathbf{H})$ is exactly the number of common walks between $\mathbf{G}$ and $\mathbf{H}$ [9]. The general formulation above allows soft matching of walks, crucial in image applications, thanks to the basis kernel $k(\ell, \ell')$.

In order to compute the walk kernels, several algorithms are available. We present a dynamic programming approach which allows nice generalization to subtree patterns for planar graphs. Alternative formulations based on matrix inversions and product graphs [16, 34] do not readily allow the same flexibility of selecting given walk lengths. Let $\mathcal{W}^p_\mathbf{G}(r)$ (resp. $\mathcal{W}^p_\mathbf{H}(s)$) denote the set of walks in $\mathbf{G}$ (resp. $\mathbf{H}$) starting at vertex r (resp. s). We denote by $k^p_\mathcal{W}(\mathbf{G}, \mathbf{H}, r, s)$ the sum defining $k^p_\mathcal{W}(\mathbf{G}, \mathbf{H})$ restricted to walks starting at r and s in each graph, i.e.,

$$k^p_\mathcal{W}(\mathbf{G}, \mathbf{H}, r, s) = \sum_{\substack{(r_1, \ldots, r_p) \in \mathcal{W}^p_\mathbf{G}(r) \\ (s_1, \ldots, s_p) \in \mathcal{W}^p_\mathbf{H}(s)}} \prod_{i=1}^{p} k(\ell_\mathbf{G}(r_i), \ell_\mathbf{H}(s_i)).$$

Let $\mathcal{N}_\mathbf{G}(r)$ (resp. $\mathcal{N}_\mathbf{H}(s)$) denote the set of neighbors of r in $\mathbf{G}$ (resp. of s in $\mathbf{H}$). The following proposition shows

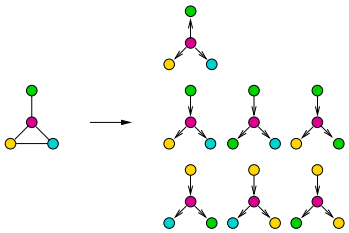

Figure 4. Examples of tree-walks from a graph. Each color represents a different label.

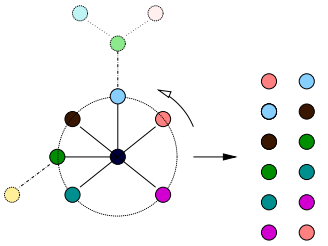

Figure 5. Enumeration of neighbor intervals of size two.

that these kernel values can be computed recursively in a dynamic programming framework.

Proposition 1 *The kernel values $k^p_{\mathcal{W}}(\mathbf{G},\mathbf{H},r,s)$ can be recursively computed through:*

$$k^p_{\mathcal{W}}(\mathbf{G},\mathbf{H},r,s) = k(\ell_{\mathbf{G}}(r),\ell_{\mathbf{H}}(s)) \sum_{\substack{r' \in \mathcal{N}_{\mathbf{G}}(r) \\ s' \in \mathcal{N}_{\mathbf{H}}(s)}} k^{p-1}_{\mathcal{W}}(\mathbf{G},\mathbf{H},r',s').$$

The recursion is initialized by $k^1_{\mathcal{W}}(\mathbf{G},\mathbf{H},r,s) = k(\ell_{\mathbf{G}}(r),\ell_{\mathbf{H}}(s))$. The final value of the kernel is then:

$$k^p_{\mathcal{W}}(\mathbf{G},\mathbf{H}) = \sum_{\substack{r \in \mathcal{V}_{\mathbf{G}} \\ s \in \mathcal{V}_{\mathbf{H}}}} k^p_{\mathcal{W}}(\mathbf{G},\mathbf{H},r,s).$$

4.2.1 Tree-walk kernels

Using walks to measure similarities between graphs might not be discriminative enough, and the natural extensions from substrings to subtrees can be carried through as well in the "walk world". A α-ary *tree-walk* (also referred to as a *subtree-pattern* in this paper) of $\mathbf{G}$ is defined as any rooted, directed α-ary tree whose vertices are vertices of $\mathbf{G}$, such that if they are neighbors in the tree pattern, they must be neighbors in $\mathbf{G}$ as well[1]. As for walks, a vertex may appear several times within the tree-walk. Yet, siblings vertices in the tree-walk must correspond to distinct vertices in the original graph. See Figure 4 for examples of tree-walks of a given graph.

If we denote by $\mathcal{T}^{p,\alpha}_{\mathbf{G}}$ (resp. $\mathcal{T}^{p,\alpha}_{\mathbf{H}}$) the set of α-ary tree patterns of $\mathbf{G}$ (resp. $\mathbf{H}$) of depth p, then, as in the definition of the walk kernel, the tree-walk kernel $k^{p,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H})$ is defined as the sum over all subtree-patterns in $\mathcal{T}_{p,\alpha}(\mathbf{G})$ and all subtree-patterns in $\mathcal{T}_{p,\alpha}(\mathbf{H})$ (sharing the same tree structure) of the products of the basis kernels between corresponding individual vertices of the subtree-patterns. Note that in the context of exact matching, investigated in [9], this kernel simply counts the number of common subtree-patterns. Note also that, as long as the basis kernel is a positive semi-definite kernel, the resulting tree-walk kernel is also a positive semi-definite kernel as well [29].

In order to derive an efficient dynamic programming formulation, we now need to restrict the set of subtrees. Indeed, if d is an upper bound on the degrees of the vertices in $\mathbf{G}$ and $\mathbf{H}$, then at each depth, the q-ary subtree-pattern may go through any subsets of size α of the set of neighbors of a given vertex, and thus the matching complexity would be $O(d^{2\alpha})$ ($O(d^{\alpha})$ for each of the two graphs). We restrict the allowed subsets of neighbors in a subtree-pattern, by requiring that these subsets are intervals for the natural cyclic ordering of the neighbors of a given vertex (this is possible because the graph is planar). See Figure 5 for an enumeration of the intervals of size $\alpha = 2$. For a given vertex r in $\mathbf{G}$ (resp. s in $\mathbf{H}$), we denote by $\mathcal{I}^{\alpha}_{\mathbf{G}}(r)$ (resp. $\mathcal{I}^{\alpha}_{\mathbf{H}}(s)$) the set of non empty intervals of length at most α around r in $\mathbf{G}$ (resp. around s in $\mathbf{H}$). In the remaining of the paper, we assume that all our subtree-patterns (referred to as tree-walks from now on) are restricted to intervals of neighbors. Let $k^{p,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H},r,s)$ denote the sum over all tree patterns starting from vertex r in $\mathbf{G}$ and s in $\mathbf{H}$.

The following proposition shows that the recursive dynamic programming formulation of Proposition 1 can be extended to tree-walk kernels:

Proposition 2 *The kernel values $k^{p,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H},r,s)$ can be recursively computed through:*

$$k^{p,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H},r,s) = k(\ell_{\mathbf{G}}(r),\ell_{\mathbf{H}}(s)) \times \sum_{\substack{I \in \mathcal{I}^{\alpha}_{\mathbf{G}}(r) \\ J \in \mathcal{I}^{\alpha}_{\mathbf{H}}(s)}} \prod_{\substack{r' \in I \\ s' \in J}} k^{p-1,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H},r',s').$$

where we sum for I and J such that card(I)=card(J).

The final kernel is then obtained as:

$$k^{p,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H}) = \sum_{\substack{r \in \mathcal{V}_{\mathbf{G}} \\ s \in \mathcal{V}_{\mathbf{H}}}} k^{p,\alpha}_{\mathcal{T}}(\mathbf{G},\mathbf{H},r,s).$$

The above equation defines a dynamic programming recursion which allows to efficiently compute the values of

[1] In this paper, we consider α-ary trees with at least one and less than α children per node, but the framework could easily be restricted to *full* α-ary trees, i.e., with exactly 0 or α children per node.

$k_{\mathcal{T}}^{p,\alpha}(\mathbf{G}, \mathbf{H}, \cdot, \cdot)$ from $p = 1$ to any desired p. It relies upon the initialization $k_{\mathcal{T}}^{1,\alpha}(\mathbf{G}, \mathbf{H}, r, s) = k(\ell_{\mathbf{G}}(r), \ell_{\mathbf{H}}(s))$ for r a segment of $\mathbf{G}$ and s a segment of $\mathbf{H}$.

Finally, note that when $\alpha = 1$ (intervals of size 1), the tree-walk kernel reduces to the walk kernel [27].

4.3. Running time complexity

Given labelled graphs $\mathbf{G}$ and $\mathbf{H}$ with $n_{\mathbf{G}}$ and $n_{\mathbf{H}}$ vertices each and maximum degrees $d_{\mathbf{G}}$ and $d_{\mathbf{H}}$, we assume that the kernel between labels $k(\ell, \ell')$ can be computed in constant time. Hence the total cost of computing $k(\ell_{\mathbf{G}}(r), \ell_{\mathbf{H}}(s))$ for all $r \in \mathcal{V}_{\mathbf{G}}$ and $s \in \mathcal{V}_{\mathbf{H}}$ is $\mathcal{O}(n_{\mathbf{G}} n_{\mathbf{H}})$.

For walk kernels, the complexity of each recursion in Proposition 1, is $\mathcal{O}(d_{\mathbf{G}} d_{\mathbf{H}})$. Thus, computation of all q-th walk kernels for $q \leqslant p$ needs $\mathcal{O}(p d_{\mathbf{G}} d_{\mathbf{H}} n_{\mathbf{G}} n_{\mathbf{H}})$ operations.

For tree-walk kernels, the complexity of each recursion of Proposition 2, is $\mathcal{O}(\alpha^2 d_{\mathbf{G}} d_{\mathbf{H}})$. Therefore, computation of all q-th α-ary tree walk kernels for $q \leqslant p$ needs $\mathcal{O}(p \alpha^2 d_{\mathbf{G}} d_{\mathbf{H}} n_{\mathbf{G}} n_{\mathbf{H}})$ operations, i.e. leading to polynomial-time complexity.

5. Engineering segmentation kernels

In order to apply the family of kernels described in Section 4 to image classification, we first need to specify the kernels between segments and then set the various parameters (depth p, arity of the tree α).

5.1. Kernel between segments

There are plenty of choices for defining relevant features for segment comparisons, ranging from median color to sophisticated local features [20]. We chose to focus on RGB color histograms since previous work [12] thoroughly investigated their relevance for image classification, when used on the whole image without any segmentation. This allows us to fairly evaluate the efficiency of our kernels to make a smart use of segmentations for classification.

Experimental results of kernels between color histograms taken as discretized probability distributions $P = (p_i)_{i=1}^N$ were given in [12]. In this paper we focus on the χ^2-kernel defined as follows: the symmetric χ^2-distance between two distributions P and Q is defined as

$$d_\chi^2(P, Q) = \sum_{j=1}^{N} \frac{(p_i - q_i)^2}{p_i + q_i},$$

and the χ^2-kernel is defined as $k_\chi(P, Q) = e^{-\mu d_\chi^2(P,Q)}$, with μ a free parameter to be tuned. Following results of [12] and [8], since this kernel is positive semi-definite, it can be used as a basis kernel. If we denote P_ℓ the histogram of colors of region labelled by ℓ, then it defines a kernel between labels as $k(\ell, \ell') = k_\chi(P_\ell, P_{\ell'})$.

Moreover, in order to avoid too strong diagonal dominance of the obtained kernel matrices (a usual issue with kernels on structured data [33]) and to ensure the positive semi-definiteness of the kernel, it is customary to include a constant term λ that controls the maximal value of $k(\ell, \ell')$. We thus use the following kernel:

$$k(\ell, \ell') = \lambda e^{-\mu d_\chi^2(P_\ell, P_{\ell'})},$$

with free parameters λ, μ.

5.2. Segments weighting scheme

It is natural to give more weight in the overall sum to massive segments than to tiny ones. Hence we incorporate this into the segment kernel as:

$$k(\ell, \ell') = \lambda A_\ell^\gamma A_{\ell'}^\gamma e^{-\mu d_\chi^2(P_\ell, P_{\ell'})},$$

where A_ℓ is the area of the corresponding region; γ is an additional free parameter in $[0, 1]$ to be tuned.

5.3. Parameters

For each kernel, the values for several parameters need to be appropriately tuned. In practical applications, this is usually done by cross-validation. While this is no major issue for problems with few parameters, in our case it is mandatory to adapt a wise strategy in order not to be overwhelmed by the parameter tuning phase. To reduce this number while gaining deeper knowledge about our kernels, we decided to fix two of them in advance. The coefficient μ in the kernel $k(\ell, \ell')$ between segment histograms is fixed to the optimal value suggested by [12]. Note that this value is different for histograms on the full image than histograms on regions: indeed, μ is taken larger for histograms on regions as colors in regions are usually more homogeneous and the matching can then be made "harder".

The coefficient λ penalizing longer walks was chosen equal to 0.75. In particular the choice of λ was guided by the behavior of the resulting Gram matrix (indeed $\lambda = 0.75$ was the best value for the least diagonally dominant Gram matrices). This leaves the following parameters to be tuned, as summed up in the table below:

Kernel	free param.	fixed param.
Histogram		μ
Walk	p	$\mu, \lambda, \alpha = 1$
Tree-walk	$p, \alpha > 1$	μ, λ
Weighted tree-walk	$p, \alpha > 1, \gamma$	μ, λ

5.4. Multiple kernel learning

In this paper we have defined several families of kernels, kernels based on histograms on the full image, walk kernels of different lengths, and tree-walk kernels of different

depths and arities. Those kernels provide access to different information regarding images and in those learning settings where heterogeneous data sources (and kernels) are available, it has proved advantageous to i) consider linear combinations of the kernels corresponding to each source, and ii) learn these linear coefficients from data, a framework referred to as *multiple kernel learning* [1]. More precisely, if we have m kernels $k_1, \ldots, k_m$, then in a single optimization problem, the optimal linear combination $\sum_{j=1}^{m} \eta_j K_j$ together with the optimal classifier are jointly learned. One attractive feature of those methods is the sparsity of the learned linear combination, i.e. only a small subset of the kernels k_j are retained. In this paper, we have used the publicly available code of [2], with $m = 100$ kernels corresponding to various settings of the parameters. See Section 6 for empirical results, both in terms of improved prediction accuracy and selection of kernels.

6. Experiments

6.1. Experimental setting

Experiments have been carried out on both `Corel14` [3] and `Coil100` [24] datasets. Our kernels were put to the test step by step, going from the less sophisticated version to the most complex one. Indeed, we compared on a multi-class classification task performances of the usual histogram kernel (**H**), the walk-based kernel (**W**), the tree-walk kernel (**TW**), the weighted-vertex tree-walk kernel (**wTW**) and the combination of the above by multiple kernel learning (**M**). We report here their performances averaged in an outer loop of 5-fold cross-validation. The hyperparameters are tuned in an inner loop in each fold by 5-fold cross-validation (see Section 6.3 for further details). `Coil100` consists in a database of 7200 images of 100 *objects in a uniform background*, with 72 images per object. Data are color images of the objects taken from different angles, with steps of 5 degrees. `Corel14` is a database of 1400 *natural images* of 14 different classes, which are usually considered much harder to classify. Each class contains 100 images, with a non-negligible proportion of outliers.

6.2. Features from segmentation

Each image's segmentation outputs a labelled graph with 100 vertices, with each vertex labelled with the RGB-color histogram within each corresponding segment. We used 16 bins per dimension, as in [12], yielding 4096-dimensional histograms. Note that we could use LAB histograms as well. Average vertex degree was around 3 for `Coil100` and 5 for `Corel14`. In other words segmentation graphs are very sparsely connected.

6.3. Free parameter selection

For the multi-class classification task, the usual SVM classifier was used in a one-versus-all setting [29]. For each family of kernels, hyper-parameters corresponding to kernel design and the SVM regularization parameter C were learned by cross-validation, with the following usual machine learning procedure: we randomly split the full dataset in 5 parts of equal size, then we consider successively each of the 5 parts as the testing set (the outer testing fold), learning being performed on the four other parts (the outer training fold). Trying out different values of the free parameters on the outer training fold, computing the prediction accuracy on the corresponding testing fold, repeating five times for each of the five outer folds, averaging performance and select the best hyper-parameter and reporting its performance leads to an optimistic estimation of the prediction performance [17]. It is preferable to consider each outer training fold, and split those into 5 equal parts, and learn the hyper-parameters using cross-validation on those inner folds, and use the resulting parameter, train on the full outer training fold with this set of hyperparameters (which might be different for each outer fold) and test on the outer testing fold. The prediction accuracies which we report (in particular in the boxplots of Figure 6) are the prediction accuracies on the outer testing folds. This two-stage approach leads to more numerous estimations of SVM parameters but provide a fair evaluation of performance.

In order to choose the values of the free parameters, we use values of free parameters on the finite grid below:

Parameter	Values
γ	$0.0, 0.2, 0.4, 0.6, 0.8$
α	$1, 2, 3$
p	$1, 2, 3, 4, 5, 6, 7, 8, 9, 10$
C	$10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}$

6.4. Results

Below are listed the respective performances in average test error rates (on the five testing outer folds) in multi-class classification of the different kernels. See Figure 6 for corresponding boxplots for the `Corel14` dataset.

	H	W	TW	wTW	M
`Coil100`	1.2%	0.8%	0.0%	0.0%	0.0%
`Corel14`	10.36%	8.52%	7.24%	6.12%	5.38%

`Coil100` **dataset** Results on `Coil100` dataset show that our framework allows to reach state-of-the art performances on simple tasks.

`Corel14` **dataset** We have compared test error rate performances of SVM-based multi-class classification with histogram kernel (**H**), walk kernel (**W**), tree-walk kernel

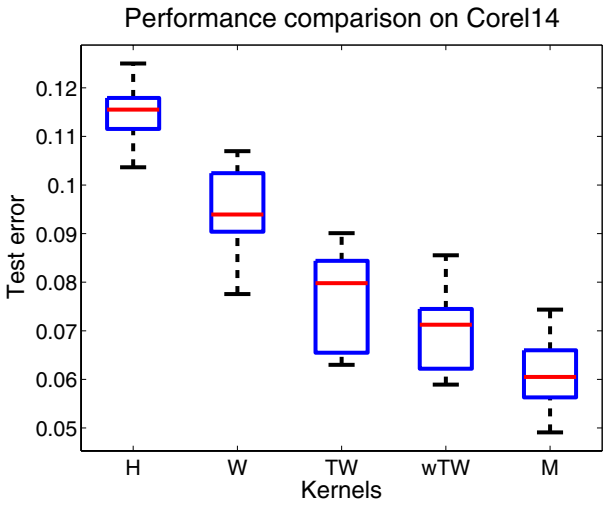

Figure 6. Test errors for supervised multi-class classification on `Corel14`, for **H**istogram, **W**alk, **T**ree-**W**alk, **w**eighted **T**ree-**W**alk, kernels and optimal **M**ultiple kernel combination.

(**TW**), and the tree-walk kernel with weighted segments (**wTW**). Our methods, i.e. **TW** and **wTW** clearly outperforms global histogram kernels and simple walk kernels. This corroborates the efficiency of tree-walk kernels to capture the topological structure of natural images. Our weighting scheme also seems to be reasonable.

Multiple kernel learning We first tried the combination of histogram kernels with walk-based kernels, which did not yield significant performance enhancement. This suggests that histogram do not carry any supplemental information over walk-based kernels: the global histogram information is implicitly retrieved in the summation process of walk-based kernels.

We ran the multiple kernel learning (MKL) algorithm of [2] with 100 kernels (corresponding to all free parameter settings in Section 6.3 except $\alpha = 2$). As seen in Figure 6, the performance increases as expected. It is also worth looking at the kernels which were selected. We here give results for one of the five outer cross-validation folds, where 5 kernels were selected.

p, α, γ	$10, 3, 0.6$	$7, 1, 0.6$	$10, 3, 0.3$	$5, 3, 0.0$	$8, 1, 0.0$
η	0.12	0.17	0.10	0.07	0.04

It is worth noting that various values of γ, α and p are selected, showing that each setting may indeed capture different discriminative information from the segmented images.

6.5. Semi-supervised classification

Kernels allow us to tackle many tasks, from unsupervised clustering to multi-class classification and manifold learning [29]. To further explore the expressiveness of our segmentation graph kernels, we give below the evolution of test error performances on `Corel14` dataset for multi-class classification with 10% labelled examples, 10% test examples, and an increasing amount ranging from 10% to 80% of unlabelled examples. Indeed, all semi-supervised algorithms derived from statistically consistent supervised ones see their test errors falling down as the number of labelled examples is increased and the number of unlabelled examples kept constant. However, such experimental convergence of the test errors as the number of labelled examples is kept fixed and the number of unlabelled ones is increased is much less systematic [4]. We used the publicly available code for the low density separation (LDS) algorithm of [4], since it reached good performances on `Coil100` image dataset.

Since we are more interested in showing the flexibility of our approach than the semi-supervised learning problem in general, we simply took as a kernel the optimal kernel learned on the whole supervised multi-classification task by the multiple kernel learning method. Although this may lead to a slight over-estimation of the prediction accuracies, this allowed us to bypass the kernel selection issue in semi-supervised classification, which still remains unclear and under active investigation. For each amount of unlabelled examples, as an outer loop we randomly selected 10 different splits into labelled and unlabelled examples. As an inner loop we optimized the hyperparameters, namely the regularization parameter C and ρ the cluster squeezing parameter (see [4] for details), by leave-one-out cross-validation. The boxplots in Figure 7 shows the variability of performances within the outer loop. Keeping in mind that the best test error performance on `Corel14` dataset of histogram kernels is around 10% for *completely supervised* multi-class classification, the results are very promising; we see that our kernel reaches this level of performance for as little as 40% unlabelled examples and 10% labelled example.

7. Conclusion

We have presented a novel framework for image analysis through tree-walk kernels defined on segmented images. While the basis kernel allows to compare local information carried by segments histograms, tree-walk kernels capture the topological similarity of images. The result is a state-of-the art method for image classification with competitive performance on natural images as well as object image datasets. Above all, our family of kernels for images paves the way to the design of kernel-based methods for tackling the large range of challenging image classification tasks.

Our work thus far has been limited to simple instances of graph kernels for images and to classification tasks. There are several natural extensions that should be taken advantage of. First, we can easily handle sharper features such as histograms of gradients as in [32] and integrate them with

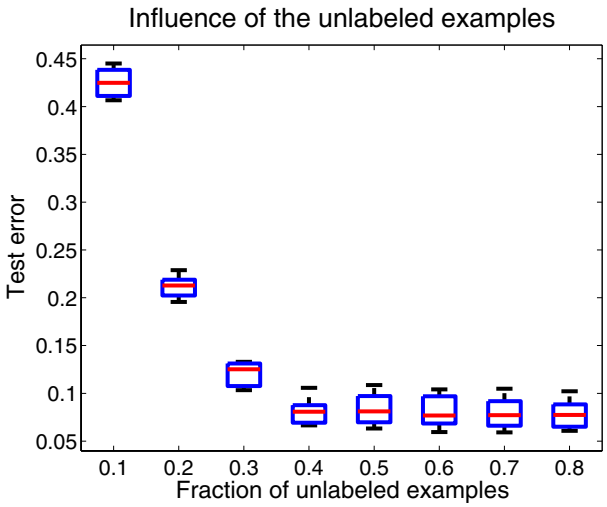

Figure 7. Test error evolution for semi-supervised multi-class classification as the number of unlabelled examples increases.

multiple kernel learning. A similar integration of heterogeneous features could be achieved with kernels that act on different information from the images, such as SIFT features [20]. Second, one could exploit recent extensions of kernels on structured data used in bioinformatics, such as the non-tottering trick [21] and gap-weighted kernel on strings [29]. Third, our kernel-based framework carries directly over clustering, semi-supervised classification, and dimensionality reduction.

References

[1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proc. ICML*, 2004.

[2] F. R. Bach, R. Thibaux, and M. I. Jordan. Computing regularization paths for learning multiple kernels. In *Adv. NIPS 17*, 2004.

[3] O. Chapelle and P. Haffner. Support vector machines for histogram-based classification. *IEEE Trans. Neural Networks*, 10(5):1055–1064, 1999.

[4] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proc. AISTATS*, 2004.

[5] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE PAMI*, 24(5):603–619, 2002.

[6] M. Cuturi, K. Fukumizu, and J.-P. Vert. Semigroup kernels on measures. *J. Mac. Learn. Research*, 6:1169–1198, 2005.

[7] R. Diestel. *Graph Theory*. Springer-Verlag, 2005.

[8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE PAMI*, 26(2):214–225, 2004.

[9] T. Gärtner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *COLT*, 2003.

[10] C. Gomila and F. Meyer. Graph based object tracking. In *Proc. ICIP*, pages 41–44, 2003.

[11] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge Univ. Press, 1997.

[12] M. Hein and O. Bousquet. Hilbertian metrics and positive-definite kernels on probability measures. In *AISTATS*, 2004.

[13] F.-J. Huang and Y. LeCun. Large-scale learning with svm and convolutional nets for generic object categorization. In *Proc. CVPR*, 2006.

[14] B. Huet, A. D. Cross, and E. R. Hancock. Graph matching for shape retrieval. In *Adv. NIPS*, 1999.

[15] T. Jebara. Images as bags of pixels. In *Proc. ICCV*, 2003.

[16] H. Kashima, K. Tsuda, and A. Inokuchi. *Kernels for Graphs*. MIT Press, 2004.

[17] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[19] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. CVPR*, 2004.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vision*, 60(2):91–110, 2004.

[21] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In *ICML*, 2004.

[22] J. Malik, S. Belongie, T. K. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. Comp. Vision*, 43(1):7–27, 2001.

[23] F. Meyer. Hierarchies of partitions and morphological segmentation. In *Scale-Space and Morphology in Computer Vision*. Springer-Verlag, 2001.

[24] S. Nene, S. Nayar, and H. Murase. Columbia object image library: Coil, 1996.

[25] M. Neuhaus and H. Bunke. Edit distance based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.

[26] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *CVPR*, 1997.

[27] J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, 2003.

[28] A. Robles-Kelly and E. Hancock. Graph edit distance from spectral seriation. *IEEE PAMI*, 27(3):365–378, 2005.

[29] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, 2004.

[30] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, 2000.

[31] F. Suard, V. Guigue, A. Rakotomamonjy, and A. Benshrair. Pedestrian detection using stereo-vision and graph kernels. In *IEEE Symposium on Intelligent Vehicule*, 2005.

[32] F. Suard, A. Rakotomamonjy, and A. Bensrhair. Object categorization using kernels combining graphs and histogram of gradients. In *Proc. ICIAR*, 2006.

[33] J.-P. Vert, H. Saigo, and T. Akutsu. *Local Alignment Kernels for Biological Sequences*. MIT Press, 2004.

[34] S. Vishwanathan, K. M. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In *Adv. NIPS*, 2007.