Nonlinear Deep Kernel Learning for Image Annotation

Mingyuan Jiu*, Hichem Sahbi

Abstract

Multiple kernel learning (MKL) is a widely used technique for kernel design. Its principle consists in learning, for a given support vector classifier, the most suitable convex (or sparse) linear combination of standard elementary kernels. However, these combinations are shallow and often powerless to capture the actual similarity between highly semantic data, especially for challenging classification tasks such as image annotation. In this paper, we redefine multiple kernels using deep multi-layer networks. In this new contribution, a deep multiple kernel is recursively defined as a multi-layered combination of nonlinear activation functions, each one involves a combination of several elementary or intermediate kernels, and results into a positive semi-definite deep kernel. We propose four different frameworks in order to learn the weights of these networks: supervised, unsupervised, kernel-based semi-supervised and Laplacian-based semi-supervised. When plugged into support vector machines (SVMs), the resulting deep kernel networks show clear gain, compared to several shallow kernels for the task of image annotation. Extensive experiments and analysis on the challenging ImageCLEF photo annotation benchmark, the COREL5k database and the Banana dataset validate the effectiveness of the proposed method.

Index Terms

Multiple kernel learning, image annotation, deep learning, semi-supervised learning

I. Introduction

With the exponential growth of social networks, huge image collections are nowadays available, and this makes their manual annotation completely out-of-hand. With this substantial growth, *automatic* solutions are currently emerging in order to annotate and search these huge image collections. Image annotation [1], [2], [3] is a widely studied topic in computer vision; it consists in assigning, for a given scene, one or multiple keywords (a.k.a concepts or categories) that best describe its visual and semantic content. These concepts may either correspond to individual objects (persons, cars, etc.) or more complex entities resulting from the interaction of multiple objects/persons into scenes (fights, cities, etc.).

Image annotation is still an open challenge in computer vision especially when contents are highly variable and when categories are taken from large (and also fine-grained) vocabularies. Standard (shallow) visual features, widely used in single object recognition, become powerless in order to capture the rich semantic content in images. Indeed, with simple visual features, images sharing the same semantic content may exhibit strong visual variability while images with a high visual similarity, may have different semantic interpretations.

In spite of these challenges, a substantial progress has been made in automatic image annotation in the last decade (see Section II). In many existing solutions, features are extracted from different sources (visual, textual, etc.), and decision criteria are built on top of these features in order to i) model intricate relationships between features and concepts, and to ii) decide about the presence of these concepts in test images. Among these models, SVMs [4], [5], are widely studied and successfully applied to image annotation. The strength of SVMs resides in their ability to model highly nonlinear decision boundaries, that separate images from different categories with a high precision, using the kernel trick. The latter makes it possible to map nonlinearly separable data from an input space to a high (possibly infinite) dimensional space where these data become linearly separable; with this mapping, a linear decision boundary is found in the high dimensional space.

The success of kernel machines (SVMs in particular) is highly dependent on the choice of kernels. The latters are defined as symmetric and positive semi-definite functions that return similarity between data as inner products involving implicit or explicit kernel maps. Relevant kernels should reserve high similarity values if data share similar content and vice-versa. Existing standard (also referred to as *elementary*) kernels include linear, polynomial, gaussian and histogram intersection (see for instance [6]). More sophisticated kernels (e.g. the spatial pyramid [7] and the convolution kernels [8]) are also designed for semi-structured data (such as interest points in images). In practice, knowing a priori which elementary or sophisticated kernel is suitable for a given classification task is not always feasible, especially when data are taken from complex (and possibly unknown) distributions.

In comparison to the aforementioned handcrafted kernels, which require domain specific knowledge, learning suitable kernels from training data [8], [9], [10] has been attracting much attention recently. Multiple kernel learning (MKL) has become a mainstream solution that learns (sparse or convex) linear combinations of elementary kernels to improve the discrimination capacity of kernels for a given classification task [9]. The strength of MKL resides in its ability to express complex similarities between data by taking into account several elementary kernels. However, standard MKL considers only shallow combinations of kernels, which turn out to be powerless to capture the "right" similarity between highly semantic and variable contents. In contrast to standard (shallow) MKL, *the purpose of this contribution, is to learn deep combinations of elementary kernels that lead to better SVM annotation performances* as shown through this paper.

In this paper, we introduce a novel method that learns deep multi-layer kernel networks for image annotation. In these networks, elementary kernels are considered as inputs to the deep networks which are recursively forwarded through intermediate and output layers. Each unit in an intermediate layer corresponds to a nonlinear combination of multiple kernels in its preceding layer; in particular, a final ("sub-optimal") kernel is obtained at the output layer. We consider in this work, a unified semi-supervised learning (SSL) framework [11] that learns these deep nonlinear combinations while maximizing the performances of image annotation. SSL models the topology of both labeled and unlabeled data resulting into better annotation performances especially for categories with scarce labeled (training) data. We implement this SSL framework using two methods: the first one (kernel-based) allows us to diffuse kernel values from labeled to unlabeled data while the second one (Laplacian-based) diffuses decision scores instead of kernel values.

Considering these issues, the main contributions of this work include: i) the extension of the two-layer (standard)

MKL to multiple layers, using well chosen activation functions that guarantee better convergence, ii) the proof that the resulting global multiple kernels are positive semi-definite and iii) a unified optimization framework, which jointly combines supervised and unsupervised learning, making it possible to build deep kernels using (few) labeled training data and (abundant) unlabeled sets, in an inductive setting[1]. The learned kernel is plugged into SVMs (as well as its Laplacian variant) in order to benefit from their well established generalization power [12] and to obtain better performances as shown through experiments using the challenging ImageCLEF benchmark [13], the COREL5k database [14] and also the Banana dataset.

This paper is organized as follows: In Section II, we first review the related work in image annotation and MKL, then, we present in Section III our main contribution: a nonlinear deep kernel network design. We present in Section IV, four learning algorithms (supervised, unsupervised, kernel-based and Laplacian-based semi-supervised) that allow us to train the deep networks while in Section V, we show the validity of our method through extensive experiments using challenging datasets. Finally, we conclude the paper in Section VI.

II. Related Work

Following the main contribution of our work, we review the related work both in image annotation and MKL.

A. Image annotation

Existing image annotation models can be categorized into three major families: generative approaches, nearest neighbor and discriminative models.

Generative models proceed by estimating the posterior probability of each concept (given feature observations) and assign concepts to images by maximizing this posterior probability [15], [16], [17], [18]. In this family, simple non-parametric approaches measure this probability by modeling cluster centroids and estimate the probability of each concept proportionally to the number of images belonging to centroids and concepts [15]. Authors in [16] present the continuous relevance model (CRM) that estimates the distribution of features in each image using a non-parametric gaussian density while authors in [17] introduce the multiple Bernoulli relevance model which calculates the distribution of keywords in a vocabulary using a Bernoulli distribution. Following the same principle, Moran and Lavrenko [18] extend CRM to sparse kernel learning using a multinomial function for count-based features; alternative techniques, model the probability of concepts using parametric approaches such as gaussian mixture models [19], [20], Latent Dirichlet allocation [21] and translation-based models [22].

In the second family of annotation methods, nearest neighbor models learn weighted combinations of "keyword absence/presence" across neighboring images [23], [24], [25]. For instance, the method in [23], transfers keywords through image neighbors which are selected using an average distance w.r.t several features. Tag Propagation (TagProp) [24] optimizes the combination of base distances across different features by maximizing the log-likelihood of keyword prediction in the training set. Verma and Jawahar [25] develop a two pass K-nearest

[1]Even though semi-supervised, the proposed global setting is inductive and makes it possible to evaluate our deep kernel network on any new pair of input data.

neighbor algorithm. Their model takes into account image-to-image and image-to-label similarities and optimizes the (non-negative) weights over base distances and features using a multi-label version of the large margin nearest neighbor algorithm [26]. In [27], a mapping between visual features and keywords is obtained using kernel canonical correlation analysis which is combined with the nearest neighbor classifier, in order to obtain image annotations.

The third family of annotation methods is based on discriminative models [28], [29], [30]. The latters learn dependencies between concepts and images; each concept is treated as an independent class and a concept-specific classifier (e.g., SVMs [29], [30], artificial neural network [28], decision tree [31], etc.) is learned to detect images that belong to that class. Considering SVM-based annotation methods[2], Goh et al. [29] propose a three-step approach based on multiple ensembles of SVMs; in the first step, each ensemble of SVMs is trained independently using a random subset from the training set. The two remaining steps fuse SVM scores sequentially within each ensemble and also through different ensembles. The method in [30] formalizes the retrieval task as a ranking problem, i.e., relevant images are ranked higher than others, given a text query; with this setting, the authors define and maximize the margin for retrieval by SVM ranking.

Other techniques enhance annotation results by modeling image-to-image relationships and by diffusing annotation results through the manifold enclosing training and test images. In these methods, training and test data are usually modeled with graphs where nodes correspond to images and edges model their pairwise interactions using Graph Laplacian operators. As a result, labels can be diffused from training (labeled) to test (unlabeled) images following a transductive setting [32], [33]. Authors in [34] build explicit kernel maps and obtain promising performances in image annotation; while this method is purely visual, others referred to context-dependent [8], [35], [36], [37], benefit from contextual cues and learn explicit context dependent kernels using the geometric structure of interest points in images as well as relational and semantic contexts taken from associated tags and metadata.

B. Multiple kernel learning

MKL has been actively studied in last decade (e.g., [9], [38], [39], [40]). Its purpose is to learn the best linear combination of elementary kernels that fits a given distribution of data [38]. Different MKL algorithms are proposed in the literature including constrained quadratic programming [9], second order cone programming with sequential minimal optimization [38], semi-infinite linear programming [39], and SimpleMKL based on mixed-norm regularization encouraging sparse kernel combinations [41]. These MKL methods define shallow kernel networks which are not sufficiently powerful in order to capture complex high level (and deep) characteristics in training and test data. Recently, other solutions were proposed to learn more complex combinations of kernels; for instance, the method in [42], embeds elementary kernels in a directed acyclic graph and the best kernels are selected through hierarchical multiple kernel learning. A nonlinear extension of MKL to polynomial kernels is also proposed in [43].

Our proposed contribution, in this paper, learns nonlinear combinations of elementary kernels, but it is not merely limited to polynomial functions. It is also related to deep learning and artificial neural networks [44], [45], [46]. Convolutional neural networks (CNNs) and their variants [47], have achieved state of the art results on many

[2]As we focus in our contribution on kernels and SVMs.

challenges in computer vision and other neighboring fields [48], [49]. CNNs are usually composed of convolution layers (combined with activation functions) and pooling layers. The final layers correspond to a set of explicit feature maps which are fully connected to the output layer (via a logistic regression function) that classifies deep features into categories. These deep architectures have shown their superiority in many classification tasks compared to shallow features [49], [50].

Following the same spirit as CNNs, we extend the standard (shallow) MKL to multiple layers. The key difference of this extension, w.r.t CNNs, resides in the implicit definition of features using the kernel trick, rather than explicit feature maps. To the best of our knowledge, this deep kernel design has not been approached in the related work for general elementary kernels, though very few attempts have been undertaken mainly for polynomial kernels. Other attempts construct deep explicit (instead of implicit) kernel maps using Kernel PCA [51]; in this method, networks are composed of several stacked layers, where the input layer is dedicated to raw features, and the subsequent hidden layers correspond to the principle component features of their preceding layers, which are obtained with KPCA. This unsupervised method learns a low dimensional manifold so it does not necessarily produce "optimal" discriminative features for classification [52]. In [53], authors use kernel functions as a priori knowledge for regularization. Cho and Saul [54] develop Arc-cosine kernels, which mimic the computation of large neural nets. Arc-cosine kernels are directly used in shallow as well as deep SVM architectures; in this architecture, features in different layers are defined using kernel PCA. However, this method is not easily extendable to other kernels. A deep multiple kernel learning is proposed by Strobl and Visweswaran [55], that learns multiple layers of kernel combinations by optimizing a span bound of the leave-one-out error, instead of hinge loss.

More recently, authors in [56] introduced the convolutional kernel networks as a combination between kernel methods and deep neural networks. This method learns approximations of kernel maps using convolutional neural networks. It is also a generalization of the hierarchical kernel descriptors proposed in [57]. A supervised learning of convolutional kernel networks has also been proposed in [58]. In contrast to this related work [56], our deep kernel networks, proposed in this paper, learn *implicit* mapping functions using nonlinear combinations of elementary kernels.

Among all the existing related work, the multi-layer MKL framework proposed by Zhuang et al. [59] is the closest to our work. Nevertheless, their network is restricted to two layers and their learning algorithm is supervised. In contrast, we extend kernel networks to multiple layers, while we still guarantee the positive semi-definiteness of the obtained deep multiple kernels. Note that our network is generic and its training is achieved using SSL [34], [60], [61]; the main assumption about SSL is that close data in a high density area of the input space are likely to share the same labels. This allows us to overcome the *scarcity* of labeled data and obtain better generalization when using usual algorithms such as standard and Laplacian SVMs. In this respect, other methods, such as Goodfellow et al. [62], propose generative adversarial networks (GAN) to create fake samples and learn other discriminative models to distinguish them from real training data. The generators and discriminators are trained alternately. Several new features and training techniques are studied in [63]. Category GAN (CatGAN) [64] are also proposed to generate class-specific samples for multiple categories, at the same time, cross-entropies from labeled data are adapted into the discriminator of CatGAN and enable semi-supervised learning. Our work is conceptually different from GAN in

the fact that the goal is to design discriminative kernel functions through deep architectures, rather than generating new data.

III. Our Deep Multiple Kernel Network

Consider a collection of ℓ labeled training samples $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{\ell}$, with $\mathbf{x}_i \in \mathbb{R}^d$ being a feature vector (for instance a d dimensional bag-of-word descriptor) and $\mathbf{y}_i$ its class label in $\{-1, +1\}$ and another collection of u unlabeled test samples $\mathcal{U} = \{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$; in practice $\ell \ll u$. Our goal is to jointly learn a deep multiple kernel and classifier f from labeled and unlabeled samples. Here f is an SVM-based classifier that predicts the class $\mathbf{y}_i \leftarrow \text{sign}[f(\mathbf{x}_i)]$ of a given sample $\mathbf{x}_i \in \mathcal{U}$. Usual algorithms jointly learn kernels and SVM-based classifiers by maximizing a margin and by using labeled samples; in this work, we also consider the unlabeled data for kernel design for a better discrimination power as shown later in experiments.

A kernel (denoted κ) is a symmetric function that provides a similarity between any two given samples [4]. When positive semi-definite, a kernel is also defined as an inner product in a high (possibly infinite) dimensional space, via a mapping function (denoted ϕ). Using the kernel trick, one can express an elementary kernel, without explicitly knowing ϕ; among these kernels, polynomial and radial basis functions are the most studied [4]. In this work, we aim to learn an implicit mapping function that recursively characterizes a nonlinear and deep combination of multiple elementary kernels.

Fig. 1 shows our deep kernel network with L layers. For each layer l and its associated unit p, a kernel domain $\big\{\kappa_p^{(l)}(\cdot,\cdot)\big\}$ is recursively defined as

$$\kappa_p^{(l)}(\cdot,\cdot) = g\big(\sum_q \mathbf{w}_{q,p}^{(l-1)}\ \kappa_q^{(l-1)}(\cdot,\cdot)\big), \tag{1}$$

where g is a nonlinear activation function chosen in order to guarantee the positive semi-definiteness of the learned deep kernels (see more details in Sections III-A, III-B). In the above equation, $q \in \{1, \ldots, n_{l-1}\}$, n_{l-1} is the number of units in layer $(l-1)$ and $\{\mathbf{w}_{q,p}^{(l-1)}\}_q$ are the (learned) weights associated to kernel $\kappa_p^{(l)}$. In particular, $\{\kappa_p^{(1)}\}_p$ are the elementary kernel functions including linear and histogram intersection kernels, etc. When $L = 2$, the architecture is shallow, and it is equivalent to the 2-layer MKL of Zhuang et al. [59]. For larger values of L, the network becomes deep. For any given pair of samples, a vector containing the values of different elementary kernels on this pair is evaluated and considered as an input to our deep network. These elementary kernel values are then forwarded to the subsequent intermediate layer resulting into n_2 multiple kernels through the nonlinear combination of the previous layer, etc. The final kernel is a highly nonlinear combination of elementary kernels.

We notice that the deep kernel network in essence is a multi-layer perceptron (MLP), with nonlinear activation functions (see details in Section III-A). The difference is that the last layer is not designed for classification, rather than to deliver a similarity value. However, we can use the classical backpropagation algorithm specific for MLP to optimize the weights in the deep kernel network. Let J denotes an objective function associated to our classification problem. More details, about choices of J, are discussed in Section IV. We assume that the computation of gradients of the objective function J w.r.t the output kernel $\kappa_1^{(L)}$ (i.e. $\frac{\partial J}{\partial \kappa_1^{(L)}(.,.)}$) is tractable. According to the chain rule, the

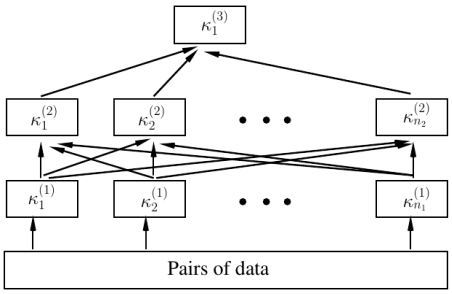


Fig. 1. The deep kernel learning architecture with three layers. The input of this network corresponds to different elementary kernels evaluated on a given pair of data.

corresponding gradients w.r.t coefficients $\mathbf{w}$ are computed, and then used to update these weights using gradient descent.

In relation to our work, Siamese networks also aim to learn similarity functions between pairs of data [65], [66], but these similarities are not guaranteed to be positive semi-definite and hence cannot always be plugged into kernel-based SVMs for classification. Moreover, the inputs of the Siamese networks should correspond to vectorial data[3] which are forwarded through different layers to obtain intermediate maps, and the similarity is computed on these maps. In contrast, our kernel networks require only implicit maps – through input kernels and their deep nonlinear combinations – and do not require the explicit generation of these maps; this is clearly beneficial especially when training tasks involve non-vectorial data such as interconnected images and graphs, where only relationships between data are available, and when explicit maps are not possible to obtain on the input data.

A. Activation functions

We consider in this paper two activation functions for $g(t)$ (in Eq. (1)); exponential $\exp(t)$ and hyperbolic $\tanh(t)$. As shown subsequently, these nonlinear activation functions increase the dimensionality of the learned (implicit) kernel maps and also guarantee their positive semi-definiteness.

More precisely, hyperbolic functions are adopted in the intermediate layers of our deep network, partly serving as normalization of the underlying kernel functions, while exponential functions are used in the remaining layers, in order to enhance the representational power of the resulting kernels; indeed, considering Taylor's expansion of the exponential function, the latter corresponds to infinite sums of monomials of increasing orders, each one maps data into higher dimensional spaces, via the Kronecker tensor product, and this clearly increases the dimensionality (and hence discrimination power) of the resulting kernel maps.

As discussed in the next section, the second advantage of these activation function choices is related to the positive semi-definiteness of the learned deep multiple kernels.

[3]This limits the usability for semi structured data, such as graphs, where only relationships/similarities between data are available.

B. Positive semi-definiteness

In this section, we show that our deep multiple kernels, in Eq. (1), are conditionally positive semi-definite (c.p.s.d). Precisely, we discuss the sufficient conditions about the choices of the elementary kernels, the learning parameters $\{\mathbf{w}_{q,p}^{(l-1)}\}_{p,q,l}$ and the activation function g that guarantee this property.

Definition 1 (conditionally p.s.d kernels): A kernel κ is conditionally p.s.d, iff $\forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, $\forall c_1, \ldots, c_n \in \mathbb{R}$ (with $\sum_{i=1}^{n} c_i = 0$), we have $\sum_{i,j} c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0$.

From the above definition, it is clear that any p.s.d kernel is also c.p.s.d; the latter is a weaker (but sufficient) condition in order to learn max margin SVMs (see for instance [4]).

Proposition 1: Provided that the input elementary kernels $\{\kappa_q^{(1)}\}_q$ are conditionally p.s.d, and $\{\mathbf{w}_{q,p}^{(l-1)}\}_{p,q,l}$ belong to the positive orthant of the parameter space; any combination defined as $g(\sum_q \mathbf{w}_{q,p}^{(l-1)} \ \kappa_q^{(l-1)})$ (with $g(t)$ equal to $\exp(t)$ or $\tanh(t)$) is also conditionally p.s.d.

Proof 1: The proof is given in Appendix.

From proposition (1), provided that i) the elementary kernels are conditionally p.s.d, ii) the activation function g preserves the c.p.s.d (as exponential and hyperbolic) and iii) weights $\{\mathbf{w}_{q,p}^{(l-1)}\}$ are positive, all the resulting multiple kernels, in Eq. (1), will also be conditionally p.s.d and can be used for max margin SVM training and classification. Other choices of activation functions in the intermediate layers are possible, for instance, the rectified linear unit function (ReLU), but the positive definiteness of the final deep kernel is not always preserved, unless all the intermediate kernel values are positive (see also comparative results in Section V-A).

IV. Learning

In this section, we introduce four different kernel learning settings depending on whether labels are available: supervised, unsupervised and semi-supervised (kernel-based as well as Laplacian-based). The semi-supervised setting relies on discriminative learning using labeled data and it also captures the topology of the manifold enclosing $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$ using abundant unlabeled data. We also consider Laplacian SVM (as a variant of the semi-supervised setting) which seeks to minimize an objective function mixing a loss term (on labeled data $\mathcal{L}$) and a regularization criterion (on SVMs applied to $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$); see also Section IV-D.

In what follows, we change slightly the notation compared to the previous section, in order to handle multiple-classes. Precisely, we fix K as the number of classes and denote $\mathbf{y}_i^k$ (with $k \in \{1, \ldots, K\}$) as the membership of $\mathbf{x}_i$; with $\mathbf{y}_i^k = +1$ if $\mathbf{x}_i$ belongs to class k and -1 otherwise. Note that the membership of a given $\mathbf{x}_i$ to a class is not exclusive. Similarly, we denote f_k as a learned classifier that assigns for a given test data $\mathbf{x}_i$ its membership to class k based on sign$[f_k(\mathbf{x}_i)]$, so a given class k (also referred to as label) belongs to a given $\mathbf{x}_i$ iff sign$[f_k(\mathbf{x}_i)] \geq 0$.

A. Supervised kernel learning

Considering a multi-class problem and given the labeled set $\mathcal{L}$, the goal is to learn a deep kernel $\kappa_p^{(L)}$ (i.e., weights $\mathbf{w}$) and SVM classifiers $\{f_k\}_k$ (i.e., parameters $\boldsymbol{\alpha}$) on top of $\kappa_p^{(L)}$ that assign labels to the unlabeled set $\mathcal{U}$. We emphasize that SVM learning is *not directly* achieved on the output of the learned deep kernel as this output is computed using a given pair (i.e., an entry of a gram matrix), while SVM learning requires the whole gram matrix

Algorithm 1: Deep Kernel Learning

Input: Initial $\mathbf{w}^{(l)}(l = 1, \ldots, L-1) > 0$, $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{\ell}$

Output: Optimal $\mathbf{w}^{(l)}(l = 1, \ldots, L-1)$, $\boldsymbol{\alpha}$

repeat

  Fix $\mathbf{w}$, compute a final output kernel $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j), \forall i, j \in 1, \ldots \ell$;

  $\boldsymbol{\alpha}^k$ is learned by SVM solver for each category;

  Fix $\boldsymbol{\alpha}$, compute the gradient of J w.r.t $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j)$;

  Update $\mathbf{w}$ using backpropagation with gradient descent steps constrained to keep $\mathbf{w} \geq 0$;

until *Convergence*;

of training data. Hence, this network has at least two advantages: on the one hand, it is less time demanding (during training) compared to a network that takes a gram matrix as input[4]. On the other hand, this makes the approach inductive, and the computation of deep multiple kernels feasible on any new (out of sample) data.

In order to optimize the network parameters $\mathbf{w}$, we use the backward information w.r.t $\kappa_p^{(L)}(\cdot, \cdot)$ from the learned SVMs. For that purpose, we learn a "one-vs-all" SVM for each class k by minimizing a global hinge loss and regularization terms

$$\min_{\mathbf{w}, \{f_k\}} \sum_{k=1}^{K} C_k \sum_{i=1}^{\ell} \max\left(0, 1 - \mathbf{y}_i^k f_k(\mathbf{x}_i)\right) + \frac{1}{2}\left\|f_k\right\|_{\mathcal{H}}^2, \tag{2}$$

here $C_k \geq 0$ controls, for each class k, the tradeoff between regularization and empirical error. According to the representer theorem [4], the dual form of Eq. (2) can be rewritten as

$$\begin{aligned} \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \sum_{i=1}^{\ell} \alpha_i^k - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i^k \alpha_j^k \mathbf{y}_i^k \mathbf{y}_j^k \quad \kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad 0 \leq \alpha_i^k \leq C_k, \ \sum_{i=1}^{\ell} \alpha_i^k \mathbf{y}_i^k = 0, \end{aligned} \tag{3}$$

and the underlying SVM classifier (associated to a class k) is $f_k(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i^k \mathbf{y}_i^k \ \kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}) + b_k$; following the KKT conditions [12], b_k is a shift that guarantees the equilibrium constraint $\sum_{i=1}^{\ell} \alpha_i^k \mathbf{y}_i^k = 0$ (in Eq. (3)).

The objective function in Eq. (3) is optimized w.r.t two parameters: $\mathbf{w}$ and $\boldsymbol{\alpha}$. Alternating optimization strategy is adopted, i.e., we fix $\mathbf{w}$ to optimize $\boldsymbol{\alpha}$, and then vice-versa. At each iteration, when $\mathbf{w}$ is fixed, $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j)$ is also fixed, and $\boldsymbol{\alpha}^k$ is optimized using an SVM solver independently for each class (LIBSVM in practice [67]). When $\boldsymbol{\alpha}^k$ is fixed, only the right-hand side term in Eq. (3) depends on $\mathbf{w}$. Therefore, the gradient of Eq. (3) w.r.t $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j)$ is $\frac{\partial J}{\partial \kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j)} = -\frac{1}{2} \sum_{k=1}^{K} \alpha_i^k \alpha_j^k \mathbf{y}_i^k \mathbf{y}_j^k$. Using the gradient of the output layer in the deep kernel network, a round of backpropagation is achieved and $\mathbf{w}$ is accordingly updated using gradient descent. The iterative procedure continues until convergence or when a maximum number of iterations is reached (see Algo. 1).

[4]We tried this variant which requires 9 hours to learn network weights and also the SVMs while our current deep network takes only 3 hours to train.

B. Unsupervised Setting

This setting aims to constrain the implicit maps of our deep kernels using a hypothesis similar to the cluster assumption [68]. For general classification problems, this assumption states that low density areas of an input space[5] are likely to correspond to decision boundaries. Analogously, we extend this principle to our kernel design as follows: *any two data belonging to different sides of a low density area of an input space are likely to produce low kernel values and vice-versa.*

More precisely, our unsupervised kernel learning aims to learn $\kappa_1^{(L)}$ using the unlabeled set $\mathcal{U}$, in order to capture the topological structure of data. As $\kappa_1^{(L)}$ corresponds to an inner product in a high dimensional Hilbert space $\mathcal{H}$, with $\phi_1^L : \mathcal{X} \rightarrow \mathcal{H}$, our unsupervised setting aims to (implicitly) learn ϕ_1^L , so that any two close/similar (resp. far) samples $\mathbf{x}_i$, $\mathbf{x}_j$ will be close (resp. far) in the mapping space $\mathcal{H}$ resulting into the following smoothness criterion

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell+u} \sum_{j \in \mathcal{N}_i} A(\mathbf{x}_i, \mathbf{x}_j) \left\| \phi_1^L(\mathbf{x}_i) - \phi_1^L(\mathbf{x}_j) \right\|_2^2, \tag{4}$$

here $\mathcal{N}_i$ corresponds to a neighborhood system, i.e., a set of neighbors of $\mathbf{x}_i$ in the input space $\mathcal{X}$ and $A(\mathbf{x}_i, \mathbf{x}_j)$ measures the visual similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathcal{X}$, taking in practice, histogram intersection. Using the kernel trick and assuming, without a loss of generality, that $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_i)$ constant ($\forall i$), Eq. (4) becomes equivalent to

$$\min_{\mathbf{w}} - \sum_{i=1}^{\ell+u} \sum_{j \in \mathcal{N}_i} A(\mathbf{x}_i, \mathbf{x}_j) \ \kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j). \tag{5}$$

Eq. (5) models the topology of data in $\mathcal{X}$ and produces deep multiple kernels whose (implicit) maps are close in $\mathcal{H}$ for neighboring data in $\mathcal{X}$. For this unsupervised setting, kernel weights $\mathbf{w}$ are independent from the SVM coefficients $\boldsymbol{\alpha}$, so the optimization is not achieved alternately as in Algo. (1), but in two steps: first gradient descent is used to optimize $\mathbf{w}$, then SVM coefficients $\boldsymbol{\alpha}$ are learned in one step, on top of the underlying deep kernel $\kappa_1^{(L)}$, using LIBSVM [67].

C. Kernel-based Semi-supervised Setting

In this section, we consider a semi-supervised kernel learning that exploits both the labeled data (in $\mathcal{L}$) and the unlabeled ones (in $\mathcal{U}$). The semi-supervised setting aims to design a suitable deep multiple kernel on $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$ by diffusing the kernel similarity from labeled data in $\mathcal{L}$ to the unlabeled ones in $\mathcal{U}$ using the smoothness criterion in Eq. (5). Hence, we define a global semi-supervised optimization criterion that combines Eq. (3) and Eq. (5) as

$$\min_{\mathbf{w}, \boldsymbol{\alpha}} \quad \lambda_1 \ J_{\mathcal{L}}(\mathbf{w}, \boldsymbol{\alpha}) + (1 - \lambda_1) \ J_{\mathcal{U}}(\mathbf{w}), \tag{6}$$

with $J_{\mathcal{L}}$, $J_{\mathcal{U}}$ being criteria in Eqs. (3), (5), respectively, and $\lambda_1 \in [0, 1]$ controls the balance between the two terms. As in Section IV-A, we use alternating optimization; again, when fixing $\boldsymbol{\alpha}$, and solving for $\mathbf{w}$, we remove the constant $\sum_{k=1}^{K} \sum_{i=1}^{\ell} \alpha_i^k$ in Eq. (3). As labels are not available in $\mathcal{U}$, only visual similarity is used to diffuse kernel values (in $\kappa_1^{(L)}$) from $\mathcal{L}$ to $\mathcal{U}$, via the smoothness criterion in Eq. (5). Finally, we optimize the parameters $\mathbf{w}$, $\boldsymbol{\alpha}$ alternately as in Algo. (1).

[5]Related to very low pdf (probability density function) values.

D. Laplacian-based Semi-supervised Setting

In Section IV-C, kernel-based semi-supervised learning combines two criteria: a standard empirical risk and a regularization term that controls the smoothness of the kernel maps. However, these two criteria act in two different spaces; the classifier outputs and the Hilbert space, and this makes their joint optimization less intuitive compared to the one introduced subsequently. Indeed, we consider in this section a different regularization criterion that controls the smoothness of the SVM scores, through the manifold enclosing training and test data. This framework is closely related to Laplacian SVMs, and its objective function is defined as

$$\begin{aligned}\min_{\mathbf{w},\{f_k\}} \sum_{k=1}^{K} C_k \sum_{i=1}^{\ell} \max\left(0, 1-\mathbf{y}_i^k f_k(\mathbf{x}_i)\right) + \frac{1}{2}\left\|f_k\right\|_{\mathcal{H}}^2 \\ + \lambda_2 \sum_{i=1}^{\ell+u} \sum_{j\in\mathcal{N}_i} A(\mathbf{x}_i,\mathbf{x}_j)(f_k(\mathbf{x}_i)-f_k(\mathbf{x}_j))^2.\end{aligned} \tag{7}$$

The above equation can be rewritten in its equivalent form using the Laplacian operator:

$$\min_{\mathbf{w},\{f_k\}} \sum_{k=1}^{K} C_k \sum_{i=1}^{\ell} \max\left(0, 1-\mathbf{y}_i^k f_k(\mathbf{x}_i)\right) + \frac{1}{2}\left\|f_k\right\|_{\mathcal{H}}^2 + \lambda_2 \mathbf{f}_k^\intercal \mathbf{L} \mathbf{f}_k, \tag{8}$$

where $\mathbf{L}$ is the graph Laplacian matrix and $\mathbf{f}_k$ is a vector that gathers the SVM membership scores of training and test data w.r.t the k^{th} class. Using the representer theorem (see for instance [61]), the solution of Eq. (8) is

$$f_k(\mathbf{x}) = \sum_{i=1}^{\ell+u} \alpha_i^k \kappa_1^{(L)}(\mathbf{x},\mathbf{x}_i). \tag{9}$$

By introducing Lagrange multipliers, Eq. (8) can further be transformed into

$$\begin{aligned}\min_{\mathbf{w}} \max_{\alpha,\beta} J = \min_{\mathbf{w}} \max_{\alpha,\beta} \sum_{k=1}^{K} \frac{1}{2} \alpha^{k\intercal}\left(\mathbf{K}_1^{(L)} + 2\lambda_2 \mathbf{K}_1^{(L)} \mathbf{L} \mathbf{K}_1^{(L)}\right)\alpha^k \\ - \alpha^{k\intercal} \mathbf{K}_1^{(L)} \mathbf{D}^\intercal \mathbf{Y}^k \beta^k + \sum_{i=1}^{\ell} \beta_i^k,\end{aligned} \tag{10}$$

where $\mathbf{K}_1^{(L)}$ is the gram matrix of our learned deep kernel, α^k is the vector of $(\ell+u)$ training parameters, $\mathbf{D} = [\mathbf{I}\ 0]$ is the $\ell \times (\ell+u)$ matrix with $\mathbf{I}$ being the $\ell \times \ell$ identity matrix and $\mathbf{Y}^k$ = diag($\mathbf{y}_1^k, \mathbf{y}_2^k, \ldots, \mathbf{y}_\ell^k$). The derivative of Eq. (10) w.r.t α^k implies

$$\alpha^k = (\mathbf{I} + 2\lambda_2 \mathbf{L} \mathbf{K}_1^{(L)})^{-1} \mathbf{D}^\intercal \mathbf{Y}^k \beta^{k*}. \tag{11}$$

By replacing (11) into (10), we get its corresponding dual problem, it can be solved by a standard SVM solver. Once solved (w.r.t β^k), α^k is calculated by Eq. (11).

Now we estimate the gradient of Eq. (10) w.r.t $\mathbf{K}_1^{(L)}$ as

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{K}_1^{(L)}} = \sum_{k=1}^{K} \frac{1}{2} \alpha^k \alpha^{k\intercal} + \lambda_2 (\mathbf{L} + \mathbf{L}^\intercal) \mathbf{K}_1^{(L)} \alpha^k \alpha^{k\intercal} \\ - \left[\alpha^k (\mathbf{Y}^k \beta^k)^\intercal\ 0_{(\ell+u)\times u}\right].\end{aligned} \tag{12}$$

Finally, we update the weights in the deep kernel using alternating optimization as shown in Algo. (1).

V. EXPERIMENTS

In this section, we compare the performance of the learned deep multiple kernels w.r.t different baseline kernels and standard MKL (i.e., with two layers) on image annotation, also known as "concept detection". Given a picture, the goal is to predict which concepts (classes) belong to that picture, i.e., concepts that describe its visual content. For this purpose, a "one-vs-all" SVM classifier is trained for each class. All the experiments are run on the recent and challenging ImageCLEF 2013 Photo Annotation benchmark [13], the COREL5k database [14] and the Banana dataset.

A. ImageCLEF 2013 database

Training and test data. The ImageCLEF Photo annotation database has three ensembles: training, development (dev) and test sets. The training set includes 250k images which are unlabeled while the dev (resp. test) set consists of 1,000 (resp 2,000) images belonging to 95 categories. As ground truth is released only on the dev set, we report image annotation performances on this set only[6]. In all the following experiments, and unless explicitly mentioned, we split the dev set into two folds of 500 samples each; one fold is used as labeled training set (i.e., $\mathcal{L}$) in order to train SVM classifiers while the remaining fold is used as unlabeled test set (i.e., $\mathcal{U}$). Depending on the setting (supervised, unsupervised or semi-supervised), either $\mathcal{L}$, $\mathcal{U}$ or both are used in order to design our deep multiple kernels (see again Sections III and IV).

Features and evaluation measures. In all these experiments, we use the bag-of-features provided by the organizers. They include four variants of SIFTs (i.e., SIFT, C-SIFT, RGB-SIFT, OPPONENT-SIFT) [69], two variants of GISTs (i.e. GIST and GIST2), LBP center features, two color features (i.e., COLORHIST and HSVHIST) [70] and GETLF features. For the choice of similarity between data, we consider a slight variant of $A(\mathbf{x}_i, \mathbf{x}_j)$ as

$$A(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \frac{1}{2}[H_{ij} + S_{ij}] & \forall i, j \in \{1, \ldots, \ell\} \\ H_{ij} & \text{otherwise,} \end{cases} \tag{13}$$

here H_{ij}, S_{ij} correspond respectively to the standard histogram intersection and the Jaccard similarity; the latter is defined as the size of "label intersection" divided by the size of "label union", i.e., $S_{ij} = \sum_{k=1}^{K}[(\mathbf{y}_i^k = +1) \wedge (\mathbf{y}_j^k = +1)] / \sum_{k=1}^{K}[(\mathbf{y}_i^k = +1) \vee (\mathbf{y}_j^k = +1)]$. With this updated A, it becomes possible to diffuse kernel values (in $\mathcal{L}$) not only through visually similar data but also through visually dissimilar ones that possibly share common labels, resulting into a better modeling of semantic similarity in $\kappa_1^{(L)}$.

We report the performances using three different measures including the F-scores (harmonic means of recall and precision) at the concept and sample levels (resp. MF-C, MF-S) and the mean Average Precision (mAP) over all categories; higher values of these scores imply better performance.

Baselines: "elementary" vs "shallow multiple" kernels. Initially, we evaluate different standard elementary kernels, including linear (lin), degree two polynomial (poly), RBF (with a scale parameter set to the average Euclidean distance between points and their neighbors) and histogram intersection (HI). Then, we train "one-versus-all" standard SVMs on $\mathcal{L}$ and we evaluate their performances on $\mathcal{U}$. The penalty parameter C_k for each

[6]Results/comparison w.r.t participants are available on dev set using the official link http://www.imageclef.org/2013/photo/annotation/results

Kernel	MF-S	MF-C	mAP
GMKL([72])	41.3	24.3	49.1
2LMKL([59])	45.0	25.8	54.0

TABLE I

THIS TABLE SHOWS THE BASELINE PERFORMANCE (IN %) FOR DIFFERENT (LINEAR AND NONLINEAR) MKL ALGORITHMS WITH SUPERVISED LEARNING.

class is chosen in $[2^{-4}, 2^{8}]$ by three fold cross-validation on the training set. MF-S, MF-C and mAP scores of these SVMs are shown in Table 1 of the supplementary material [71] for different elementary kernels and different visual features. We observe that linear kernels applied to high dimensional feature vectors (e.g. bag of 5000 codewords for C-SIFT features) are competitive against histogram intersection kernels; this results from the linear separability of data when using reasonably high dimensional input spaces (such as the 5000 codewords for C-SIFT features); thereby the linear kernel is relatively performant as a baseline.

We also train "one-versus-all" Laplacian SVMs (on $\mathcal{L}$, $\mathcal{U}$) and evaluate their performances on $\mathcal{U}$ for all the elementary kernels. The best λ_2 is also chosen for each class in $[2^{-10}, 2^{4}]$ by three fold cross-validation while C_k is set as mentioned earlier for standard SVMs. The results of these laplacian SVMs are shown in Table 2 of the supplementary material [71]; we observe that Laplacian SVMs improve the performance for most of the elementary kernels compared to standard SVMs.

In order to find the best combination of elementary kernels that improves these initial baseline results, we first train (shallow) multiple kernels (i.e., standard MKL). For that purpose, we use 40 elementary kernels (resulting from the combination of 4 kernels $\times$ 10 visual features) and we learn both linear and nonlinear MKLs. For the linear setting, we use the generalized multiple kernel learning approach, referred to as GMKL [72], as a variant of linear MKL. For the nonlinear setting, we use the 2-layer MKL (2LMKL) of Zhuang et al. [59]. Note that only supervised learning is employed in these two experiments. We observe in table I a gain of shallow MKL compared to the results of single elementary kernels, especially on the MF-S and MF-C scores, which now reach 45% and 25% while the MF-S, MF-C performances of the best elementary kernel in the supervised learning does not exceed 40% and 22.1% respectively. We also observe that nonlinear 2LMKL improves MF-C and mAP slightly over GMKL.

Shallow vs deep multiple kernels. Following the best baseline results described earlier, we consider in the following only nonlinear combinations of kernels. We examine three issues: the number of layers in our deep network, the performances obtained when using supervised, unsupervised and semi-supervised settings, and also the performance when additional unlabeled data are involved into kernel learning.

Firstly, we investigate the impact of the number of layers in the deep network on the performance of supervised learning (described in section IV); 2 layers correspond to 2LMKL while 3-7 layers correspond to its deep version. For any given pair of data, we use again 40 elementary kernel values as input to our deep network and its output corresponds to the global kernel. For 3-layer network, we tested eight architectures of different numbers of hidden units (i.e., intermediate kernels) for supervised learning as shown in Table 3 of the supplementary material [71]. We

Activation function	Framework	MF-S	MF-C	mAP
exp [59], [73]	3-layer KL	46.4	29.6	58.0
	4-layer KL	27.8	15.5	31.8
ReLU	3-layer KL	46.3	27.8	55.3
	4-layer KL	45.0	25.8	53.8
tanh	3-layer KL	46.0	29.5	55.8
	4-layer KL	46.6	29.6	56.3

TABLE II

THIS TABLE SHOWS THE PERFORMANCE (IN %) OF THREE ACTIVATION FUNCTIONS IN THE HIDDEN LAYER ON IMAGECLEF2013 (USING SUPERVISED SETTING).

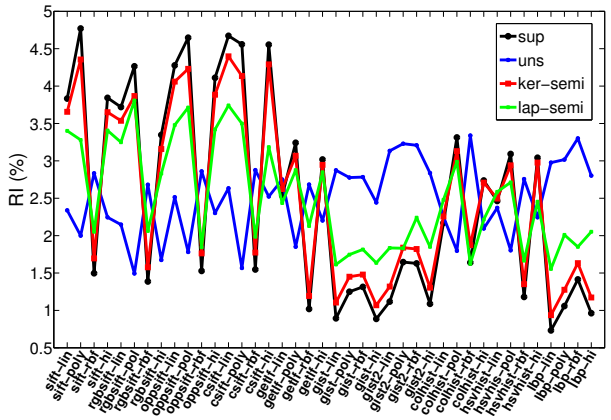


Fig. 2. This diagram shows the relative importance of input units w.r.t four settings: supervised (in black), unsupervised (in blue), kernel-based SSL (in red) and Laplacian SSL (in green). The x-axis corresponds to different combinations of elementary kernels+features, written as "feature type-kernel type", for instance, "sift-lin" stands for linear kernel + SIFT features.

found that they have comparable MF-S and mAP results, but the one with 80 hidden units improves the MF-C score from 26.7% to 29.5%. Therefore in the following experiments, a 3-layer network with 80 hidden units is adopted; similarly, the hidden layers for other deep networks have the same number of units (i.e., 80 units). Each entry in $\mathbf{w}$ is initialized randomly in $[0, 0.01]$. We have also evaluated different activation functions in the intermediate layers with 80 hidden units: ReLU, exponential (also adopted in the 2LMKL network [59]) and $\tanh$. The weights are constrained to be positive in order to preserve the positive definiteness of the obtained kernels. We observe, in Tab. II, that the exponential activation function has better mAP with 3-layers, and it is not always convergent with 4-layers, we also observe a similar behavior with ReLU. In contrast, the $\tanh$ activation function shows stable performances and preserves convergence when the network becomes deeper and this is due to its saturation property. In the following experiments, and unless explicitly mentioned, the $\tanh$ function is adopted for the hidden layers.

Table III shows performances of different learning settings for different numbers of network layers. In these

Architecture	Method	λ_1	MF-S	MF-C	mAP
2-layer KL	Supervised	1	45.0	25.8	54.0
	Unsupervised	0	44.4	28.7	54.0
	Kernel semi	10^{-5}	44.4	28.8	54.0
	Laplacian semi	-	46.8	28.6	58.9
3-layer KL	Supervised	1	46.0	29.5	55.8
	Unsupervised	0	45.5	29.4	55.0
	Kernel semi	10^{-1}	46.2	30.0	55.7
	Laplacian semi	-	**47.8**	**30.0**	**58.6**
4-layer KL	Supervised	1	46.6	29.6	56.3
	Unsupervised	0	28.8	25.1	47.7
	Kernel semi	10^{-1}	**46.6**	**31.0**	**55.9**
	Laplacian semi	-	45.5	28.5	58.4
5-layer KL	Supervised	1	46.5	29.3	56.2
	Unsupervised	0	23.8	21.3	39.9
	Kernel semi	0.9	46.1	30.6	55.6
	Laplacian semi	-	46.6	28.1	57.3
6-layer KL	Supervised	1	46.3	29.4	56.3
	Unsupervised	0	21.3	19.3	35.3
	Kernel semi	0.9	46.4	30.8	55.7
	Laplacian semi	-	46.2	27.6	56.3
7-layer KL	Supervised	1	46.5	29.7	56.1
	Unsupervised	0	19.7	15.7	21.2
	Kernel semi	0.9	46.6	30.7	55.8
	Laplacian semi	-	46.3	27.6	55.7

TABLE III

THIS TABLE SHOWS THE IMPACT OF THE NUMBER OF LAYERS AND THE SETTINGS OF DEEP LEARNING (SUPERVISED, UNSUPERVISED, KERNEL-BASED SSL, LAPLACIAN SSL) ON THE PERFORMANCE USING IMAGECLEF2013.

results, λ_1 is set by grid search; and it is common to all the classes as the smoothness term in kernel-based SSL does not involve the final SVM. However, as λ_2 affects the final Laplacian SVM, different λ_2 values (in $[2^{-10}, 2^4]$) are considered for different classes using three fold cross-validation.

Secondly, we investigate the behavior of the two semi-supervised settings of our deep kernel learning (in Section IV). Recall that we show results only on the dev set as the ground truth is released for this set only, and we split the dev set into two subsets of equal sizes: the labeled training set $\mathcal{L}$ and the unlabeled test set $\mathcal{U}$. Note that

Method	$\lvert\mathcal{U}\rvert$	MF-S	MF-C	mAP
Unsupervised	500	45.5	29.4	55.0
	1500	45.1	30.4	54.3
	2500	45.4	30.7	54.1
	3500	44.5	30.7	53.7
	4500	44.5	30.5	53.7
Kernel semi	500	46.6	31.0	55.9
	1500	46.4	31.2	55.7
	2500	47.0	31.6	55.6
	3500	46.5	**31.8**	55.3
	4500	46.7	31.7	55.7
Laplacian semi	500	47.8	30.0	58.6
	1500	50.0	29.1	**59.7**
	2500	**50.2**	27.5	58.7
	3500	49.1	27.0	58.0
	4500	48.9	26.8	57.3

TABLE IV

THIS TABLE SHOWS THE IMPACT (WHEN EXTRA DATA IN $\mathcal{U}$ ARE INVOLVED INTO KERNEL LEARNING IN UNSUPERVISED, KERNEL-BASED SSL, LAPLACIAN SSL) ON THE PERFORMANCE OF IMAGECLEF2013 DATABASE.

supervised setting uses only $\mathcal{L}$ for kernel design while the unsupervised and semi-supervised settings also rely on $\mathcal{U}$ (which can also be augmented by adding extra data from the 250k images of the unlabeled training set as shown later).

Table III shows the behavior of kernel-based SSL (in Eq. (6)) for different values of λ_1. It is clear that $\lambda_1 = 1$ corresponds to the supervised setting while $\lambda_1 = 0$ corresponds to the unsupervised one. We observe that: for supervised setting, globally deep networks produce better performances compared to shallow ones (with 2 layers) and these performances stabilize with 4 layers. For unsupervised kernel learning (with 2-layers), globally, it provides better performance compared to supervised setting and a 3-layer network is comparable for both, this is mainly due to the smoothness criterion that effectively diffuses similarity through the manifold enclosing data in $\mathcal{X}$; however, this behavior is no longer observed for the deeper networks as the complexity makes kernel learning difficult without

the use of labeled data, and the learned kernel becomes almost flat when only Eq. (4) is optimized.

A better performance is observed for intermediate values of λ_1 for kernel-based SSL; indeed, performances increase from 2-layer to 4-layer networks, and then stabilize on deeper networks. However, on 4-layer network, MF-S value is comparable to supervised setting and mAP smaller. This results from the fact that kernel map regularization in kernel-based SSL is not explicitly (directly) related to the final SVM outputs.

Finally, the best performance is obtained when using Laplacian SSL (with a 3-layer network), and this overtakes kernel-based SSL; hence, it becomes clear that the impact of the smoothness term in kernel-based SSL is not sufficient to correctly diffuse labels to unlabeled data. This also shows that Laplacian SSL (when combined with the learned deep kernel) is able to better diffuse similarity and labels from few labeled to abundant unlabeled data, thereby making the learned deep kernel more relevant and also less sensitive to the scarceness of labeled training data. We also observe that the performance of Laplacian SSL first increase from 2-layer to 3-layer networks, and then decrease when the number of hidden layers is larger. We conjecture that more labeled data is necessary when the network becomes more complex and deeper.

Thirdly, we evaluate the performance when more unlabeled data are added in our deep kernel learning. For that purpose, we randomly sample up to 4,000 unlabeled images from the whole unlabeled set of ImageCLEF (which includes 250k images); resulting from the hardness of the task in ImageCLEF, the visual content of these 4,000 images is very different from the content of training/dev set (provided by challenge organizers). We evaluate the performance of unsupervised, kernel-based SSL and Laplacian SSL, each using the best configuration from Tab. III: 3-layer network for unsupervised and Laplacian SSL settings and 4-layer network for kernel-based SSL. Tab. IV shows a noticeable gain when more unlabeled data are used in kernel design. We observe that: i) for unsupervised setting, MF-S and mAP scores slightly decrease as the cardinality of the unlabeled set increases; ii) for kernel-based SSL, MF-C increases and reaches 31.8% while MF-S and mAP scores remain stable; iii) for Laplacian SSL, mAP increases to 59.7% and MF-S to 50.2%, while MF-C slightly decreases to 29.1%. We have also investigated that larger unlabeled sets (more than 4,000 images) does not show improvement and this is mainly due to serious dilution of the labeled data in learning, especially for MF-C values.

Sensitivity analysis. We analyze here the relative importance of the input units associated to the combination of 40 elementary kernels+features. The goal is to check which combinations of elementary kernels+features influence the most the output of the deep network. The relative importance [77] of a given unit q (belonging to the input layer) with respect to the unit o of the output layer ($o = 1$ in our network) for the 3-layer network[7] is defined as

$$\mathrm{RI}_{qo} = \frac{\sum_{p=1}^{n} \left(\mathbf{w}_{q,p}^{(1)} \mathbf{w}_{p,o}^{(2)} / \sum_{k=1}^{m} \mathbf{w}_{k,p}^{(1)} \right)}{\sum_{q'=1}^{m} \sum_{p=1}^{n} \left(\mathbf{w}_{q',p}^{(1)} \mathbf{w}_{p,o}^{(2)} / \sum_{k=1}^{m} \mathbf{w}_{k,p}^{(1)} \right)}, \tag{14}$$

where subscripts q, q', k refer to input units, p is a hidden unit and m, n denote the number of units in input and hidden layers respectively (in practice $m = 40$ and $n = 80$ as discussed earlier). In the above equation, $\mathbf{w}_{q,p}^{(1)}$ is the weight from unit q to unit p; similarly, we define all the other weights in Eq. (14) which measure the relative importance of a given input unit by considering the weights from all the hidden units.

[7]This can easily be generalized to more than 3 layers.

Manner	Model	Framework	Classifier	Kernel learning	Context	**R**	**P**	$\mathbf{N}_+$
Generative	CRM [16]	Shallow	-	No	No	19	16	107
	InfNet [74]	Shallow	-	No	No	24	17	112
	MRFA [75]	Shallow	-	No	Yes	36	31	172
	SKL-CRM [18]	Shallow	-	Yes	No	46	39	184
Discriminative	JEC-15 [23]	Shallow	KNN	No	Yes (learned)	33	28	140
	TagPop σML [24]	Shallow	KNN	No	Yes (learned)	42	33	160
	2PKNN [25]	Shallow	KNN	No	Yes (learned)	40	39	177
	2PKNN-ML [25]	Shallow	KNN	No	Yes (learned)	46	44	191
	wTKML [34]	Shallow	SVM	Yes	Yes (fix)	42	21	173
	KSVM-VT [76]	Shallow	SVM	No	Yes (fix)	42	32	179
	SDMKL(proposed)	Deep	SVM	Yes	Yes (fix)	38	25	158
	LDMKL(proposed)	Deep	SVM	Yes	Yes (fix)	44	29	179

TABLE V

THE PERFORMANCE ON COREL5K DATABASE FOR DIFFERENT MODELS. "SDMKL" STANDS FOR KERNEL-BASED SSL ON 3-LAYER NETWORK WHILE "LDMKL" STANDS FOR LAPLACIAN-BASED SSL ON 3-LAYER NETWORK.

Fig. 2 shows the results (in %) of four learning algorithms. There are three issues worth to be noted: firstly, the RI values between unsupervised learning (in blue) and the other three learning settings have different distributions. The RI values between elementary kernels are roughly uniform, partly because the objective of unsupervised learning is to maximize kernel values for neighbors, regardless of the types of kernels and features. However, for the other three learning settings, the RI of kernels based on SIFT-type features are prominent among all these combinations. These features capture more appearance information, and are therefore more relevant for classification. Secondly, the RI values of kernels based on SIFT-type features for kernel-based SSL and Laplacian-based SSL are lower, compared the supervised setting, for instance, "SIFT-POLY"(4.36% and 3.28% vs 4.77%), and "CSIFT-HI"(4.29% and 3.18% vs 4.55%). At the same time, other non-prominent kernels are boosted when using SSL, as a result, SSL prefers different and complementary sources of features+kernels and obtain the best performances. Finally, Laplacian-based SSL has a less contrasted distribution (between different elementary kernels) compared to kernel-based SSL; it further reduces the prominence of SIFT-type features and boosts other features. Class-by-class comparison of MF-C scores and annotation examples are shown in Figs. (1) and (2) of the supplementary material [71].

B. COREL5k database

Training and test data. Initially introduced in [14], this dataset has become an important benchmark to evaluate the performance of different annotation approaches. It contains 5,000 images, among which 499 images are used for testing (i.e., $\mathcal{U}$) while the rest are used for training (i.e., $\mathcal{L}$). Each image is annotated with up to five keywords (taken from a vocabulary of 200 keywords).

Features and evaluation measures. We adopt 15 types of INRIA features for COREL5k database [24], including GIST, 6 color histograms for RGB, LAB, HSV in two spatial layouts, 8 bag-of-features based on SIFT and robust

hue descriptor densely or sparsely extracted in two spatial layouts for each image. We report the performances using the mean precision $\mathbf{P}$ and recall $\mathbf{R}$ over keywords. $\mathbf{N}_+$ is also computed and corresponds to the number of keywords with non-zero recall value. Following the standard protocol, each image is annotated with up to 5 keywords.

Results and discussion. As in the ImageCLEF dataset, we consider 4 elementary kernels for each feature, i.e., linear, order two polynomial, RBF (with a scale parameter set to the average distance between data) and histogram intersection; in total, 60 different elementary kernels are taken as inputs to the deep network. Since the 3-layer deep network achieves the best performances in the ImageCLEF database, we adopt the same architecture on the COREL5k database with a slight difference in the number of units (equal to 120 instead of 80) in the hidden layers. The weights in the network are also randomly initialized in $[0, 0.01]$ and the similarity $A(\mathbf{x}_i, \mathbf{x}_j)$ between data is computed by the heat kernel where its width is set to the mean distance among neighbors.

We train an ensemble of "one-versus-all" standard (as well as Laplacian) SVM classifiers for each category; each classifier is learned using all the positive data and a random subset of negative data whose cardinality is equal to the number of positive data. This sampling makes it possible to handle the severe imbalanced class distributions due to the abundance of negative data. The average decision scores from all the learned classifiers are regarded as final scores for a given class. Tab. V shows the performance of kernel-based and Laplacian-based SSL (when combined with our learned deep kernel network) and a comparison w.r.t other models, including different generative and discriminative models.

We observe from Tab. V that our learned nonlinear deep kernels – when plugged into SVMs – are very competitive compared to related state-of-the-art models (SKL-CRM for generative models and 2PKNN-ML for discriminative ones). In these comparative methods, SKL-CRM is an improved version of the CRM model [18] where an optimal joint probability distribution is estimated using a greedy "kernel-feature" alignment algorithm; the latter defines for each feature a distance function using a set of candidate kernels, including χ^2 and multinomial kernels. One can further improve performances of SKL-CRM by plugging our deep kernel framework into it in order to learn "optimal" kernel-feature alignments.

The other family of comparative methods – KNN based – learns context (neighborhood relationships of each sample) by optimizing weights for different combinations of distances and features; in particular, 2PKNN-ML [25] is a metric learning version of the 2PKNN model, which is a two-step variant of KNN. More precisely, it learns "image-to-label" relationships as well as "image-to-image" similarities in two steps; besides "image-to-label" relationships, our semi-supervised deep kernel learning also exploits "image-to-image" similarities using graphs. However, node connections in our graphs are pre-defined while in [25] these weights are learned. We believe that learning these connections (as a future work) will further enhance our performances.

Note also that these KNN-based methods (including the Logistic Discriminant in TagPop σML model) are designed to handle imbalanced class distributions whereas most of the comparative SVM-based algorithms are not; indeed, in these SVM settings, classes are processed independently and imbalanced class distributions are not rectified. For the sake of compatibility with these comparative SVM methods, and besides computational issues related to training, we have chosen to report performances using a similar (simple) version of SVM. It is clear that rectifying the effect

Method	LDKL [78]	SDMKL	LDMKL
Accuracy (in %)	88.67	91.07	91.05

TABLE VI

THIS TABLE SHOWS A COMPARISON OF LDKL, KERNEL-BASED SSL AND LAPLACIAN SSL OF 3-LAYER DEEP KERNEL NETWORK ON BANANA DATASET.

of imbalanced class distributions – using ensemble training with resampling – will bring an extra substantial gain in performances[8].

Among the related SVM-based methods, KSVM-VT (Kernalized SVM with Variable Tolerance) [76] introduces a tolerance parameter in the hinge loss to handle incomplete-labeling, label-ambiguity as well as label-overlap. However their performances are limited as kernels are fixed (not learned). Finally, the wTKML model [34] is the most related work; it learns transductive kernel maps on labeled and unlabeled data. Under this very comparable setting, we clearly observe in Tab. V that our Laplacian-based SSL method learns more discriminative kernels and shows a clear gain compared to wTKML.

C. Banana dataset

In order to further compare our deep kernel network w.r.t the related work, mainly local deep kernel learning (LDKL [78]), we use the challenging Banana dataset and the experimental protocol in [78]. As introduced in [78], the LDKL defines a composite 2-layer kernel network as the product of two kernels that capture global and local similarity between data using tree-structured features; note that [78] considers primal SVM optimization on local deep features trained on a tree-structure.

The Banana dataset includes 1,000 training and 4,300 test samples; each one is endowed with a binary class label. We feed the input of our deep kernel network with 21 RBF kernels associated to 21 scale factors ranging from $2^{-10}M$ to $2^{10}M$; here M is the average Euclidean distance between all samples in the Banana set. In these experiments, we adopt a 3-layer deep kernel network whose intermediate layer contains 42 units, and we set the hyper-parameters (C_k, λ_1 and λ_2) on a random validation set including 20% of the training data. The classification accuracy of kernel-based SSL, Laplacian SSL with the 3-layer kernel network are illustrated in Tab. VI together with the accuracy of LDKL (as shown in [78]). From this table, we observe that Laplacian-based and kernel-based SSL behave similarly and both clearly outperform LDKL.

VI. CONCLUSION

We introduced in the paper a novel method for multiple kernel design based on deep networks. The strength of this method resides in the deep architecture that models similarity between data using multiple features and kernels combined, in a recursive way, resulting into better performance as shown for the challenging task of image

[8]This has been tested for other training tasks on imbalanced datasets using simple (and fast) kernels. However, due to the interleave between kernel learning and SVM learning, retraining with resampling becomes computationally expensive.

annotation. We also investigated several settings of our deep kernel networks: supervised, unsupervised, kernel-based and Laplacian-based semi-supervised. The latter shows superior performances due to the integration of abundant unlabeled data in kernel design.

ACKNOWLEDGMENT

This work was supported in part by a grant from the research agency ANR (Agence Nationale de la Recherche) under the MLVIS project, ANR-11-BS02-0017.

APPENDIX

PROOF OF PROPOSITION 1

Details of the first part of the proof, based on recursion, are omitted and result from the application of definition (1) to $\sum_q \mathbf{w}_{q,p}^{(l-1)}\ \kappa_q^{(l-1)}$ (for different values of l) while considering $\{\kappa_q^{(1)}\}_q$ c.p.s.d. Now we show the second part of the proof (i.e., if κ is c.p.s.d, then $g(\kappa)$ is also c.p.s.d).

Considering Taylor's expansion

i) For $g(t) = \exp(t)$, we have

$$\exp(\kappa(\mathbf{x}, \mathbf{x}')) = \sum_{n=0}^{\infty} \frac{\kappa(\mathbf{x}, \mathbf{x}')^n}{n!},$$

as the positive powers of $\kappa(\mathbf{x}, \mathbf{x}')$ preserve the c.p.s.d, the linear combination (with positive weights $\frac{1}{n!}$) in $\exp(\kappa)$ also preserves the c.p.s.d.

ii) For $g(t) = \tanh(t)$, one may write

$$\tanh(\kappa(\mathbf{x}, \mathbf{x}')) = \sum_{n=1}^{\infty} \frac{B_{2n} 2^{2n}(2^{2n}-1)}{(2n)!} [\kappa(\mathbf{x}, \mathbf{x}')]^{2n-1},$$

here $\{B_{2n}\}_n$ are the Bernoulli numbers. As $\{B_{2n}\}_n$ are not always positive, one cannot directly apply the closure of c.p.s.d w.r.t powers and linear combination (as in $\exp(\kappa)$), in order to infer that $\tanh(\kappa)$ is c.p.s.d. We consider instead definition (1); $\tanh(\kappa)$ is c.p.s.d iff $\forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$, $\forall c_1, \ldots, c_n \in \mathbb{R}$, (with $\sum_i c_i = 0$), we have $\sum_{i,j} c_i c_j \tanh(\kappa(\mathbf{x}_i, \mathbf{x}_j)) \geq 0$.

We rewrite $\sum_{i,j} c_i c_j \tanh(\kappa(\mathbf{x}_i, \mathbf{x}_j))$ as

$$\sum_{i,j} c_i c_j \sum_{n=1}^{\infty} S_{2n-1}\ \kappa(\mathbf{x}_i, \mathbf{x}_j)^{4n-3} + S_{2n}\ \kappa(\mathbf{x}_i, \mathbf{x}_j)^{4n-1}, \tag{15}$$

with

$$S_k = \frac{B_{2k} 2^{2k}(2^{2k}-1)}{(2k)!}. \tag{16}$$

Without loss of generality, one may easily normalize κ in $[0, 1]$ to guarantee $\kappa(\mathbf{x}_i, \mathbf{x}_j)^{4n-3} \geq \kappa(\mathbf{x}_i, \mathbf{x}_j)^{4n-1}$, hence

$$(15) \quad \geq \quad \sum_{n=1}^{\infty} [S_{2n-1} + S_{2n}] \sum_{i,j} c_i c_j\ \kappa(\mathbf{x}_i, \mathbf{x}_j)^{4n-3} \geq 0, \tag{17}$$

which results from $S_{2n-1} > 0$, $S_{2n} < 0$, $S_{2n-1} + S_{2n} > 0$ (according to the properties of Bernoulli numbers) and also $\sum_{i,j} c_i c_j \kappa(.,.)^{4n-3} \geq 0$ (as $\kappa(.,.)^{4n-3}$ is c.p.s.d).□

REFERENCES

[1] Y. Liu, D. Zhang, and G. Lu, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40(1), pp. 262–282, 2007.

[2] D. Zhang, M. Islam, and G. Lu, "A review on automatic image annotation techniques," *Pattern Recognition*, vol. 45, pp. 346–362, 2012.

[3] C. Cusano, G. Ciocca, and S. R., "Image annotation using svm," in *Proceedings of the SIPE*, 2004, pp. 330–338.

[4] J. Shawe-Taylor and N. Cristianini, "Kernel methods for pattern analysis," *Cambriage University Press*, 2004.

[5] S. Lyu, "Mercer kernels for object recognition with local features," in *2005 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, p. 223.

[6] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[7] K. Grauman and T. Darrell, "The pyramid match kernel: Efficient learning with sets of features," *Journal of Machine Learning Research*, vol. 8, pp. 725–760, 2007.

[8] H. Sahbi, J.-Y. Audibert, and R. Keriven, "Context-dependent kernels for object classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, pp. 699–708, 2011.

[9] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[10] C. Corinna, M. Mehryar, and R. Afshin, "Two-stage learning kernel algorithms," in *ICML*, 2010, pp. 239–246.

[11] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-supervised learning*, ser. Adaptive computation and machine learning. MIT Press, 2006.

[12] V. N. Vapnik, "Statistical learning theory," *A Wiley Interscience Publication*, 1998.

[13] M. Villegas, R. Paredes, and T. B., "Overview of the imageclef 2013 scalable concept image annotation subtask," 2013.

[14] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," in *ECCV*, 2002.

[15] A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang, "Image classification for content-based indexing," *IEEE Transactions on Image Processing*, vol. 10, pp. 117–130, 2001.

[16] V. Lavrenko, R. Manmatha, and J. Jeon, "A model for learning the semantics of pictures," in *NIPS*, 2003.

[17] S. L. Feng, R. Manmatha, and V. Lavrenko, "Multiple bernoulli relevance models for image and video annotation," in *CVPR*, 2004.

[18] S. Moran and V. Lavrenko, "Sparse kernel learning for image annotation," in *ICMR*, 2014.

[19] J. Li and J. Wang, "Real-time computerized annotation of pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 985–1002, 2008.

[20] G. Carneiro, A. Chan, P. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 394–410, 2007.

[21] K. Barnard, P. Duygulu, N. de Freitas, and D. Forsyth, "Matching words and pictures," *Journal of Machine Learning Research*, vol. 3, 2003.

[22] O. Yakhnenko and V. Honavar, "Annotating images and image objects using a hierarchical dirichlet process model," in *MDM*, 2008.

[23] A. Makadia, V. Pavlovic, and S. Kumar, "A new baseline for image annotation," in *ECCV*, 2008, pp. 316–329.

[24] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, "Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation," in *ICCV*, 2009, pp. 316–329.

[25] Y. Verma and C. Jawahar, "Image annotation using metric learning in semantic neighbourhoods," in *ECCV*, 2012, pp. 836–849.

[26] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[27] L. Ballan, T. Uricchio, L. Seidenari, and A. Bimbo, "A cross-media model for automatic image annotation," in *ICMR*, 2014.

[28] K. Kuroda and M. Hagiwara, "An image retrieval system by impression words and specific object names-iris," *Neurocomputing*, vol. 43, pp. 3–14, 2002.

[29] K. Goh, E. Chang, and B. Li, "Using one-class and two-class svms for multiclass image annotation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 1333–1346, 2005.

[30] D. Grangier and S. Bengio, "A discriminative kernel-based approach to rank images from text queries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1371–1384, 2008.

[31] R. Wong and C. Leung, "Automatic semantic annotation of real-world web images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1933–1944, 2008.

[32] W. Tong and R. Jin, "Semi-supervised learning by mixed label propagation," in *Proceedings of the 22nd National Conference on Artificial Intelligence*, 2007, pp. 651–656.

[33] H. Sahbi, J. Audibert, and R. Keriven, "Graph-cut transducers for relevance feedback in content based image. retrieval," in *ICCV*, 2007.

[34] P. Vo and H. Sahbi, "Transductive kernel map learning and its application to image annotation," in *ICCV*, 2007.

[35] X. Li, C. Snoek, and M. Worring, "Learning tag relevance by neighbor voting for social image retrieval," in *MIR conference*, 2008.

[36] H. Sahbi and X. Li, "Context-based support vector machines for interconnected image annotation," in *ACCV*, 2011, pp. 214–227.

[37] H. Sahbi, "Imageclef annotation with explicit context-aware kernel maps," *Int J Multimed Info Retr*, vol. 4, pp. 113–128, 2015.

[38] F. Bach, G. Lanckriet, and M. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *ICML*, 2004.

[39] S. Sonnenburg, G. Rätsch, C. Schafer, and B. Schölkopf, "Large scale multiple kernel learning," *Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.

[40] M. Gönen and E. Alpaydin, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.

[41] A. Rakotomamonjy, F. Bach, C. S., and G. Yves, "Simplemkl," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

[42] F. Bach, "Exploring large feature spaces with hierarchical multiple kernel learning," in *NIPS*, 2009, pp. 1–9.

[43] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels," in *NIPS*, 2009.

[44] Y. LeCun, L. Botto, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[46] Y. Bengio and P. Lamblin, "Greedy layer-wise training of deep networks," in *NIPS*, 2007.

[47] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *ECCV*, 2010.

[48] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[50] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[51] B. Scholkopf, A. Smola, and K. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 1319, pp. 1299–1319, 1998.

[52] K. Q. Weinberger, F. Sha, and L. K. Saul, "Learning a kernel matrix for nonlinear dimensionality reduction," in *ICML*, 2014.

[53] K. Yu, W. Xu, and Y. Gong, "Deep learning with kernel regularization for visual recognition," in *NIPS*, 2009, pp. 1889–1896.

[54] Y. Cho and L. Saul, "Kernel methods for deep learning," in *NIPS*, 2009, pp. 1–9.

[55] E. V. Strobl and S. Visweswaran, "Deep multiple kernel learning," in *ICMLA*, 2013, pp. 414–417.

[56] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," in *NIPS*, 2014.

[57] L. Bo, R. X., and D. Fox, "Kernel descriptors for visual recognition," in *NIPS*, 2010.

[58] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," in *NIPS*, 2016.

[59] J. Zhuang, I. Tsang, and S. Hoi, "Two-layer multiple kernel learning," in *ICML*, 2011, pp. 909–917.

[60] T. Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, 1999.

[61] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[62] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[63] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *ArXiv e-prints*, 2016.

[64] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *ArXiv e-prints*, 2015.

[65] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015, pp. 815–823.

[66] K. Zagoruyko, "Learning to compare image patches via convolutional neural networks," in *CVPR*, 2015, pp. 4353–4361.

[67] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[68] H. Narayanan and M. Belkin, "On the relation between low density separation, spectral clustering and graph cuts," *NIPS*, 2006.

[69] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[70] J. Sánchez-Oro, S. Montalvo, A. S. Montemayor, J. J. Pantrigo, A. Duarte, V. Fresno, and R. Martínez, "URJC&UNED at ImageCLEF 2013 Photo Annotation Task," in *CLEF 2013 Evaluation Labs and Workshop, Online Working Notes*, 2013.

[71] "https://www.dropbox.com/s/j7sw6kjim90fzfb/SuppTIP.pdf?dl=0".

[72] M. Varma and B. Babu, "More generality in efficient multiple kernel learning," in *ICML*, 2009.

[73] M. Jiu and H. Sahbi, "Semi supervised deep kernel design for image annotation," in *ICASSP*, 2015.

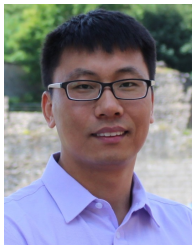
[74] D. Metzler and R. Manmatha, "A inference network approach to image retrieval," in *CIVR*, 2004.

[75] Y. Xiang, X. Zhou, T.-S. Chua, and C.-W. Ngo, "A revisit of generative model for automatic image annotation using markov random fields," in *CVPR*, 2009.

[76] Y. Verma and C. Jawahar, "Exploring svm for image annotation in presence of confusing labels," in *BMVC*, 2013.

[77] G. Garson, "Interpreting neural network connection weights," *Artificial Intelligence Expert*, vol. 6(4), pp. 46–51, 1991.

[78] C. Jose, P. Goyal, and A. P., "Local deep kernel learning for efficient non-linear svm prediction," in *ICML*, 2013.



Mingyuan Jiu received his MSc degree in Information and Communication Engineering in 2010, from Harbin Institute of Technology, China, and his PhD degree in Information in 2014 from Institut National des Sciences Appliquées de Lyon, France. He was a postdoctoral research fellow at Télécom ParisTech, Université Paris Saclay from July 2014 to December 2015. He is currently a postdoctoral researcher at École Normale Supérieure de Lyon, France. His research interests include image processing and analysis, pattern recognition, machine learning.

Hichem Sahbi received his MSc degree in theoretical computer science from the University of Paris Sud, Orsay, France, and his PhD in computer vision and machine learning from INRIA/Versailles University, France, in 1999 and 2003, respectively. From 2003 to 2006 he was a research associate first at the Fraunhofer Institute in Darmstadt, Germany, and then at the Machine Intelligence Laboratory at Cambridge University, UK. From 2006 to 2007, he was a senior research associate at the l'École des Ponts ParisTech, Paris, France. Since 2007, he has been a CNRS CR1 associate professor at Télécom ParisTech and currently at l'UPMC, Paris. His research interests include statistical machine learning, kernel and graph based inference, computer vision, and image retrieval.