SEMI SUPERVISED DEEP KERNEL DESIGN FOR IMAGE ANNOTATION

Mingyuan Jiu, Hichem Sahbi

CNRS LTCI lab, Telecom ParisTech

ABSTRACT

It is commonly agreed that the success of support vector machines (SVMs), is highly dependent on the choice of particular similarity functions referred to as kernels. The latter are usually handcrafted or designed using appropriate optimization schemes. Multiple kernel learning (MKL) is one possible scheme that designs kernels as sparse or convex linear combinations of existing elementary functions. However, this results into shallow kernels, which are powerless to capture the right similarity between data, especially when content of these data is highly semantic.

In this paper, we redefine multiple kernels using a deep architecture. In this new formulation, a global kernel is learned as a multi-layered linear combination of activation functions, each one involves a combination of several elementary or intermediate functions on multiple features. We propose three different settings to learn the weights of these kernel combinations; supervised, unsupervised and semi-supervised. When plugged into SVMs, the resulting deep multiple kernels show a gain, compared to shallow kernels, for the challenging task of image annotation using the ImageCLEF benchmark.

***Index Terms*—** Deep kernel learning, multiple kernel learning, support vector machines.

1. INTRODUCTION

Kernel learning has been successfully applied to a variety of tasks in pattern recognition and obtained promising performances (see for instance [1, 2, 3]). The family of algorithms based on kernels has been well studied [4], for example, SVM, kernel PCA and support vector regression. In a typical use of a kernel-based algorithm the crucial step is how to select an appropriate kernel; the latter, defined as an inner product in a high-dimensional Hilbert space, should reserve high values when data share similar content and vice-versa. Instead of handcrafted kernels, which usually require domain-specific knowledge, much effort has recently been undertaken in order to design suitable kernels from training data [5, 6, 7].

Several algorithms have been proposed in the literature to learn combinations of kernels. MKL is one of them, which aims to learn a (sparse or convex) linear combination of elementary kernels in order to find which elementary (or combination of) kernels are the most suitable for a given distribution of data [8, 9]. A variant of this method, known as hierarchical multiple learning [10] attempts to learn a linear combination of an exponential number of basic kernels, represented as a product of sums. At the same time, non-linear combination of polynomial kernels are also studied in [11].

Recently, artificial neural networks regained great attention by the breakthrough of deep learning methods [12, 13, 14]. Indeed, deep neural network architectures, for instance, Convolutional Neural Networks (CNNs) [15, 16], have achieved state-of-the-art results on many challenges (see for instance [17, 18]). Among these works, very few of them attempted to learn kernels in a deep way. The method in Yu et al. [19] takes advantage of an oracle kernel function and combines it into deep neural network as a regularization term. Authors in [20] develop a family of Arc-cosine kernels that mimic the computation of large neural nets and several layers are stacked to compose multiple kernel machines, and in [21], authors design explicit kernel maps from raw image data using a convolutional kernel network[1]. Zhuang et al. [22] propose multi-layer MKL framework, but restrict the learning to only two layers. A recent extension, in Strobl and Visweswaran [23], learns multiple layers of kernels by optimizing a span bound of the leave-one-out error. Kernel PCA [24] is also used to learn deep kernel maps, where the outputs (i.e. principle component features of KPCA) at a given layer are used as inputs of KPCA at the next layer. Note that this method is not discriminative and it is mainly driven by the learning of a low dimensional manifold so it does not necessarily produce suitable features for classification [25].

In this paper, we introduce a novel deep kernel design framework for classification, which considers supervised discriminative learning while at the same time it captures the topological structure of input data. The two main contributions of this work include (i) an extension, motivated by Zhuang et al. [22], that upgrades the two-layer MKL method in [22] to multiple layers, and (ii) a unified optimization framework, which jointly combines supervised and unsupervised learning. This makes it possible to build deep kernels using (few) labeled training data and (abundant) unlabeled test data in a transductive setting, resulting into better generalization performances as corroborated through experiments.

This work was supported in part by a grant from the Research Agency ANR (Agence Nationale de la Recherche) under the MLVIS project.

[1] In contrast to [21], our proposed method considers implicit kernel maps.

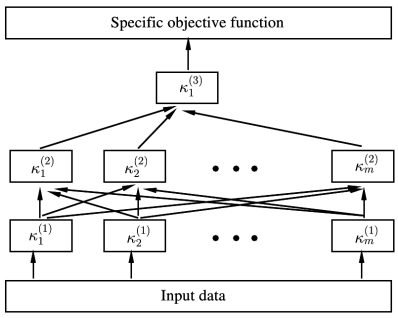

Fig. 1. The deep kernel learning architecture.

2. OUR DEEP MULTIPLE KERNEL NETWORK

For a given binary classification problem, we 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 $\{-1, +1\}$. We also consider a 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 learn a deep kernel function together with a classifier* $f : \mathcal{U} \rightarrow \{-1, +1\}$ that predicts the class $\mathbf{y}_i \leftarrow \text{sign}[f(\mathbf{x}_i)]$ of a given sample $\mathbf{x}_i \in \mathcal{U}$. In this paper, f corresponds to a kernel-based SVM. A kernel (denoted κ) is a symmetric function that defines a similarity between any two given samples [4]. When a given kernel is positive semi-definite, it can be expressed as an inner product in a high (possibly infinite) dimensional space. Many standard kernels are used in the literature including the linear, the polynomial and the radial basis functions [4].

Considering a deep kernel architecture (see Fig. 1), and given a layer l and its associated unit p, we define $\left\{\kappa_p^{(l)}(\cdot,\cdot) = g\big(\sum_q \mathbf{w}_{p,q}^{(l-1)} \ \kappa_q^{(l-1)}(\cdot,\cdot)\big)\right\}$ as the set of all possible kernels obtained by linearly and recursively combining kernels from layer $(l-1)$; here $l = 2, \ldots, L$ (for a fixed L), g is a non-linear activation function[2], $q \in \{1, \ldots, n_{l-1}\}$, n_{l-1} is the number of units in layer $(l-1)$ and $\{\mathbf{w}_{p,q}^{(l)}\}_q$ are the (learned) weights associated to kernel $\kappa_p^{(l)}$. In particular, $\{\kappa_p^{(1)}\}_p$ correspond to elementary kernels including gaussian and linear. When $L = 2$, the resulting single layer architecture corresponds to standard *(shallow) multiple kernels* while larger values of L correspond to a stack of several layers defining *(deep) multiple kernels*. As will be shown later in this paper, reasonably deep architectures are more effective and lead to better generalization for classification tasks. Figure 1 shows our deep multiple kernel architecture; apart from bottom and top layers, all the intermediate layers correspond to multiple kernels. In this architecture, n_1 elementary kernels are precomputed from labeled (and also unlabeled) data and fed as input to this deep network. The subsequent hidden layer produces n_2 multiple kernels through the combination of the previous layer, etc. The final kernel is a highly non-linear combination of elementary kernels.

[2]In practice, we adopt the same choice as Zhuang et al. [22] and we choose the exponential function $g(t) = \exp(t)$.

Algorithm 1: Deep Kernel Learning

Input: Initial $\mathbf{w}^{(l)}(l = 1, \ldots, L-1)$, and $\frac{\partial J}{\partial \kappa_1^{(L)}(.,.)}$

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

repeat
  for $l = L : 2$ **do**
    Compute gradient w.r.t $\mathbf{w}_{p,q}^{(l-1)}$:
    $\frac{\partial J}{\mathbf{w}_{p,q}^{(l-1)}} = \sum_{i,j} \frac{\partial J}{\partial \kappa_p^{(l)}(\mathbf{x}_i,\mathbf{x}_j)} . \kappa_p^{(l)}(\mathbf{x}_i, \mathbf{x}_j) . \kappa_q^{(l-1)}(\mathbf{x}_i, \mathbf{x}_j)$;
    Compute the sensitivity of $\kappa_q^{(l-1)}(\mathbf{x}_i, \mathbf{x}_j)$:
    $\frac{\partial J}{\partial \kappa_q^{(l-1)}(\mathbf{x}_i,\mathbf{x}_j)} = \frac{\partial J}{\partial \kappa_p^{(l)}(\mathbf{x}_i,\mathbf{x}_j)} . \kappa_p^{(l)}(\mathbf{x}_i, \mathbf{x}_j) . \mathbf{w}_{p,q}^{(l-1)}$;
  end
  Update the weights $\mathbf{w}_{p,q}^{(l)}$:
  $\mathbf{w}_{p,q}^{(l)} = \mathbf{w}_{p,q}^{(l)} - \beta \frac{\partial J}{\mathbf{w}_{p,q}^{(l)}} \quad l = 1, \ldots, L-1$. % The learning rate β is set by Armijo's rule, we constrain it be not too large.
until *Convergence*;

Let J denote an objective function associated to a given training and classification task; details and variants of J are discussed in Section 3. In practice, the weights of our deep multiple kernels are optimized using gradient descent. A greedy procedure (based on the chain rule and starting from the top of the deep network) is used to make gradient computation tractable[3]. More precisely, we assume that the gradients of an objective function J w.r.t the output kernel $\kappa_1^{(L)}$ (i.e. $\frac{\partial J}{\partial \kappa_1^{(L)}(.,.)}$) can be calculated. According to the chain rule, the corresponding gradients w.r.t $\mathbf{w}$ can be computed and used to update these weights using gradient descent. Algorithm 1 summarizes the whole training procedure.

3. LEARNING

In this section, we introduce three different learning settings: supervised, unsupervised and semi-supervised kernel learning. The latter makes it possible to learn deep multiple kernels not only from labeled data but also from unlabeled data using their local topological structure. 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 we 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 $\text{sign}[f_k(\mathbf{x}_i)]$, so a given class k (also referred to as label) belongs to a given $\mathbf{x}_i$ iff $\text{sign}[f_k(\mathbf{x}_i)] \geq 0$.

3.1. Supervised Deep Multiple Kernel Learning

Considering 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.,

[3]Generally speaking, any objective function could be used, as long as its derivatives can be calculated.

parameters $\boldsymbol{\alpha}$) on the top of $\kappa_p^{(L)}$ that assign labels to the unlabeled set $\mathcal{U}$. To learn $\kappa_p^{(L)}$, we use the backward information from the learned SVMs. This is achieved by optimizing the following criterion, that mixes hinge loss with regularization

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

here $C_k \geq 0$ controls, for each class k, the trade-off between regularization and empirical error and $\|\mathbf{w}^{(l)}\|_F^2$ is a regularization term that keeps the weights in $\mathbf{w}$ as flat as possible. According to the representer theorem [4], Eq. 1 can be rewritten, in its dual form

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

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 [26], b_k is a shift that guarantees the equilibrium constraint $\sum_{i=1}^{\ell} \alpha_i^k \mathbf{y}_i^k = 0$ (in Eq. 2). The objective function in (2) is optimized w.r.t two parameters: $\mathbf{w}$ and $\boldsymbol{\alpha}$. It is clear that (2) is not convex jointly w.r.t the two parameter sets (and w.r.t $\mathbf{w}$ alone) and also difficult to optimize if one considers the optimization problem w.r.t to $\mathbf{w}$ and $\boldsymbol{\alpha}$ simultaneously. Hence, we adopt alternating optimization [22], i.e., we first fix $\mathbf{w}$ and learn $\boldsymbol{\alpha}$, then vice-versa. This iterative process continues until convergence (i.e., when $\mathbf{w}$ and learn $\boldsymbol{\alpha}$ remain stable from one iteration to another). For each iteration, we use LIBSVM [27] in order to optimize $\boldsymbol{\alpha}$ and gradient descent for $\mathbf{w}$ (see again Algorithm 1).

3.2. Unsupervised Setting

The unsupervised setting aims to learn a kernel $\kappa_1^{(L)}$ from data without labels using the unlabeled sets $\mathcal{U}$ (and possibly by ignoring labels in $\mathcal{L}$). Considering $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$ and ϕ_1^L as a kernel map (either implicit or explicit) that takes samples from the input space $\mathcal{X}$ to a high dimensional mapping space $\mathcal{H}$ (with $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi_1^L(\mathbf{x}_i), \phi_1^L(\mathbf{x}_j) \rangle$), we define a smoothness criterion that guarantees similar maps in $\mathcal{H}$ for neighboring data in $\mathcal{X}$. More precisely, two close (resp. far) samples $\mathbf{x}_i, \mathbf{x}_j$ in the input space $\mathcal{X}$ should also be close (resp. far) in the mapping space $\mathcal{H}$. In order to implement this smoothness assumption, we define the following criterion

$$\min_{\mathbf{w}} \sum_{i=1}^{\ell+u} \sum_{j \in \mathcal{N}_i} \left\|\phi_1^L(\mathbf{x}_i) - \phi_1^L(\mathbf{x}_j)\right\|_2^2 - \sum_{j \notin \mathcal{N}_i} \left\|\phi_1^L(\mathbf{x}_i) - \phi_1^L(\mathbf{x}_j)\right\|_2^2, \tag{3}$$

here $\mathcal{N}_i$ corresponds to a neighborhood system, i.e., a set of neighbors of $\mathbf{x}_i$ in the input space $\mathcal{X}$. Using the kernel trick and assuming, without a loss of generality, that $\kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_i)$ constant ($\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$), we rewrite criterion in Eq. (3) as

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{\ell+u} \sum_{j \in \mathcal{N}_i} \kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{j \in \mathcal{N}_i} \kappa_1^{(L)}(\mathbf{x}_i, \mathbf{x}_j) + \sum_{l=1}^{L} \|\mathbf{w}^{(l)}\|_F^2, \tag{4}$$

where a regularization term is also added. In Eq. 4, the second term seeks to maximize kernel values for neighboring data in $\mathcal{X}$ while the first term aims to minimize kernel values for non-neighbors. Hence, we implement smoothness assumption in order to model the topology of data in the input space $\mathcal{X}$, and this leads to better modeling of similarity in the deep kernel $\kappa_1^{(L)}$ as shown later in experiments. Finally, gradients of Eq. 4, and weights $\mathbf{w}$ are updated as shown in Algorithm 1.

3.3. Semi-supervised Setting

For a better conditioning of the objective function (2), we consider in this section a semi-supervised setting to learn the deep multiple kernel $\kappa_1^{(L)}$. This makes it possible to design smooth kernel maps and to diffuse the kernel similarity from labeled data in $\mathcal{L}$ to unlabeled data in $\mathcal{U}$ through the smoothness criterion in (4). Hence, we define a global semi-supervised optimization criterion that combines (2) and (4) as

$$\min_{\mathbf{w},\boldsymbol{\alpha}} \ (1-\lambda)\, \mathbf{E}_{\mathcal{L}}(\mathbf{w}, \boldsymbol{\alpha}) + \lambda\, \mathbf{E}_{\mathcal{U}}(\mathbf{w}), \tag{5}$$

with $\mathbf{E}_{\mathcal{L}}$, $\mathbf{E}_{\mathcal{U}}$ being criteria in (2), (4) respectively and $\lambda \in [0, 1]$ balances between the two terms. Again, we use Algorithm 1 together with LIBSVM to optimize $\mathbf{w}$ and $\boldsymbol{\alpha}$.

4. EXPERIMENTAL RESULTS

In this section, we compare the performance of our multilayer deep kernel learning (using the three different settings of section 3) w.r.t different baseline kernels and the standard MKL (i.e., with a single layer). We plugged all these learned kernels into SVMs to evaluate their performance. The targeted task is image annotation also known as "concept detection"; given a picture, the goal is to predict which concepts (classes) are present into that picture. For this purpose, we trained "one-versus-all" SVM classifiers (as shown in section 3) for each concept to detect concepts in test images. We run these experiments on the recent and challenging ImageCLEF Photo Annotation database [28] (see Fig. 2).

As ground truth is only released for the dev set, we use the latter to evaluate the annotation performances[4]. This set consists in 1,000 images belonging to 95 categories, for all these images, we use ten visual features provided by the organizers (see first row in Tab 1). We run experiments, twice using 2 splits of the dev set; each split corresponds to 2 random folds of 500 samples, the first fold is used as labeled training set[5] (i.e., $\mathcal{L}$) and the remaining fold as an unlabeled test set (i.e.,

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

[5]Note that the size of labeled sets for single concepts $\ll 500$.

	SIFT	C-SIFT	RGB-SIFT	OPP-SIFT	COLORHIST	GETLF	GIST	GIST2	HSVHIST	LBP
lin	41.7/22.9/51.0	43.2/23.7/52.3	42.4/**24.0**/51.0	42.0/23.4/51.3	36.9/19.8/39.6	33.9/18.5/34.3	33.5/16.2/34.8	34.8/17.6/37.0	32.0/16.8/32.2	35.3/20.5/38.6
pol	43.3/18.3/52.9	43.5/16.2/53.8	43.9/19.0/53.3	43.2/18.5/53.1	40.3/20.8/47.3	37.9/19.7/44.4	34.9/17.3/40.1	36.6/18.4/42.3	35.4/16.7/41.2	36.8/20.8/42.8
RBF	42.2/21.5/51.9	43.3/23.2/53.3	42.6/22.2/52.4	42.2/22.8/53.0	39.3/20.1/46.8	36.0/19.5/40.9	36.9/18.6/43.8	39.1/20.3/46.0	34.1/16.6/39.4	36.7/18.9/43.6
HI	43.2/21.8/52.4	**44.1**/23.4/**54.1**	43.1/22.8/52.9	43.9/23.0/53.4	41.9/24.4/50.0	37.0/20.2/43.3	36.7/17.5/43.3	38.6/18.8/45.2	36.4/17.9/43.4	38.1/17.9/45.0

Table 1. This table shows the baseline performance (in %) for different basic kernels; triple scores (./././) correspond respectively to MF-S, MF-C and MAP performances.

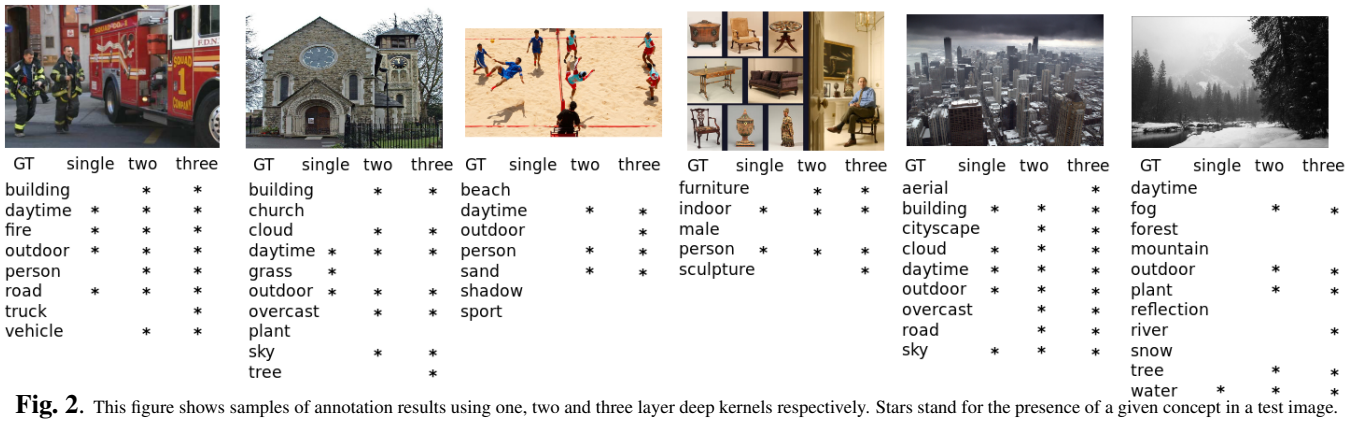

Fig. 2. This figure shows samples of annotation results using one, two and three layer deep kernels respectively. Stars stand for the presence of a given concept in a test image.

$\mathcal{U}$); the goal is then to predict classes of data in $\mathcal{U}$. We report performances using different measures including the F-scores (harmonic means of recall and precision) at the concept and sample levels (resp. MF-C, MF-S) and Mean Average Precision (MAP) averaged over the 2 splits and all samples; higher values of these scores imply better performance.

Table 1 shows the MF-S, MF-C and MAP scores for different standard kernels including linear (lin), degree two polynomial (poly), histogram intersection (HI) and RBF (with a scale parameter set to the average Euclidean distance between points and their neighbors). These results are shown for different visual features (taken from the ImageCLEF challenge). From this table, the best baseline results are obtained with HI which globally shows a gain compared to other kernels.

Kernel	MF-S	MF-C	MAP
2-layer KL	46.19	28.59	**58.40**
3-layer KL	**46.39**	**29.56**	58.00
4-layer KL	27.80	15.45	31.78

λ	MF-S	MF-C	MAP
0	46.39	29.56	58.00
0.3	**46.78**	29.49	**58.47**
0.6	46.44	29.87	58.16
0.9	45.91	29.37	57.92
0.99	46.31	**30.26**	58.14
1	46.31	29.18	57.33

Table 2. (Left) Performance measures (in %) for supervised deep kernel learning with different layers. (Right) Performance measures (in %) for semi-supervised learning w.r.t different values of λ (using 3-layer network).

In order to find a combination of elementary kernels (in Tab 1) that further improve performances, all 40 elementary kernels are fed into our deep network. We first apply our supervised learning framework (in section 3.1) with different number of layers; 2 layers correspond to the standard shallow multiple kernel learning while 3-4 layers correspond to the deeper version of our kernel. Note that each setting has 40 input elementary kernels and 1 global (learned) output kernel, 1 hidden layer of 20 units (i.e., intermediate kernels) for the 3-layer network and 2 hidden layers (each one with 20 units) for the 4-layer network. Table 2, left, shows performances for different number of layers; it is clear form these results that the learned deep multiple kernels have a gain compared to both baseline individual kernels in Tab 1 and the shallow multiple kernels (Tab 2, left, first row). However, the performance dramatically decreases when using 4-layer network and this is due to the lack of labeled training data in the dev set w.r.t complexity (number of parameters) of 4-layer network. Finally, we examine in Tab 2, right, the behavior of our semi-supervised deep kernel learning approach (in Eq. 5), with 3-layers, for different values of λ; it is clear that $\lambda = 0$ corresponds to the supervised setting while $\lambda = 1$ corresponds to the unsupervised setting. From this table the best behavior is obtained for intermediate values of λ and this shows that the semi-supervised setting is able to diffuse the similarity from labeled to unlabeled data, thereby making the learned deep kernel more relevant and also less sensitive to the scarceness of labeled training data.

5. CONCLUSION

We introduced in this paper a novel deep kernel learning framework based on three settings: supervised, unsupervised and semi-supervised. The latter takes benefit from (few) labeled training data and (abundant) unlabeled test sets to model the topological structure of data, resulting into a more relevant modeling of similarity in the learned kernels. Experiments conducted using the challenging ImageCLEF annotation task show a gain of our deep kernels compared to usual elementary kernels as well as standard (shallow) multiple kernels and competitive results w.r.t challenge scores. As a future work, we are currently investigating the application of this method to other classification challenges and tasks.

6. REFERENCES

[1] B. Caputo, C. Wallraven, and M.-E. Nilsback, "Object categorization via local kernels," in *ICPR*, 2004.

[2] S. Lyu, "Mercer kernels for object recognition with local features," in *CVPR*, 2005.

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

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

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

[6] C. Corinna, M. Mehryar, and R. Afshin, "Two-stage learning kernel algorithms," in *ICML*, 2010.

[7] H. Sahbi, J.-Y. Audibert, and R. Keriven, "Context-dependent kernels for object classification," *PAMI*, vol. 33, pp. 699–708, 2011.

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

[9] J. Zhuang, J. Wang, S. Hoi, and X Lan, "Unsupervised multiple kernel learning," in *ACML*, 2011, pp. 129–144.

[10] F. Bach, "Exploring large feature spaces with hierarchical multiple kernel learning," in *NIPS*, 2009.

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

[12] G. Hinton, S. Osindero, and Y-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, 2006.

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

[14] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, 2009.

[15] 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.

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

[17] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *PAMI*, vol. 35, no. 8, pp. 1915–1929, 2013.

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

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

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

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

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

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

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

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

[26] C.J.C Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 43, pp. 1–43, 1998.

[27] 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.

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