DEEP KERNEL MAP NETWORKS FOR IMAGE ANNOTATION

Mingyuan Jiu, Hichem Sahbi

LTCI, CNRS, Télécom ParisTech, Université Paris-Saclay, France

ABSTRACT

Deep multiple kernel learning is a powerful technique that selects and *deeply* combines multiple elementary kernels in order to provide the best performance on a given classification task. This technique, particularly effective, becomes intractable when handling large scale datasets; indeed, multiple nonlinear kernel combinations are time and memory demanding.

In this paper, we propose a new framework that significantly reduces the complexity of deep multiple kernels. Given a deep kernel network (DKN), our method designs its equivalent deep map network (DMN), using multi-layer explicit maps that approximate the initial DKN with a high precision. When combined with support vector machines, the design of DMN preserves high classification accuracy compared to its underlying DKN while being (at least) an order of magnitude faster. Experiments conducted on the challenging ImageCLEF2013 annotation benchmark, show that the proposed DMN is indeed effective and highly efficient.

Index Terms— Deep kernel networks, deep map networks, multiple kernel learning, image annotation.

1. INTRODUCTION

A large variety of kernel-based algorithms has been successfully applied to different machine learning tasks including kernel PCA for dimensionality reduction [1] and support vector machines (SVMs) for pattern recognition [2, 3, 4, 5]; in these methods, the choice of kernels is highly determinant. Kernels are symmetric functions that return similarity between data and define inner products in high (possibly infinite) dimensional spaces [6, 7]. Different kernels are proposed in the literature including linear, Gaussian and histogram intersection. In practice, knowing a priori which kernel (or a combination of kernels) is suitable for a given classification task is not always possible and a variety of machine learning techniques are introduced for kernel selection (see for instance [8, 9, 10, 11, 12]).

Multiple kernel learning (MKL) [8, 13, 14] is one of these techniques that learns a (sparse or convex) linear combination of elementary kernels that best fits a given training set. Extensions of MKL to nonlinear combinations are also proposed in the literature and turn out to be more effective: for instance Cortes et al. [15] propose nonlinear combination of polynomial kernels while Zhuang et al. [16] introduce a multi-layer nonlinear MKL framework, but restrict it to two-layers only. In this method, nonlinear (exponential) activation functions are applied to the resulting combination of basic kernels. In a more recent work, Jiu and Sahbi [17] extend this method to more than two layers using a semi-supervised setting. In all these multiple kernel learning algorithms, there is a common drawback: kernel evaluation is computationally expensive; considering a dataset with N images, the computational complexity of evaluating the gram matrix associated to a deep kernel network (with L layers) reaches $O(LN^2)$. Hence, for reasonably large values of L and N, this evaluation process becomes clearly intractable.

It is widely known that positive semi-definite kernels can be expressed as inner products involving (explicit or implicit) maps in high dimensional Hilbert spaces. It is also known that i) the map of a linear combination of kernels results from the concatenation of their maps, and ii) the map of a product of kernels is obtained using the Kronecker tensor product of their maps. Consequently, kernel maps associated to linear and polynomials kernels can be explicitly and exactly defined. With explicit kernel maps, one may use very efficient SVM learning algorithms based on stochastic gradient descent [18] in order to handle very large scale training datasets, without evaluating huge gram matrices and without solving intractable quadratic programming problems. However, other (and also effective) kernels, such as histogram intersection, may not have exact explicit kernel maps, and different algorithms have been proposed to approach these maps using, for instance, the Nyström expansion cite Williams2001. Other methods, e.g. Rahimi an Recht [19], consider instead random Fourier feature map for stationary kernels by randomly sampling the Fourier spectrum, while Li et al. [20] extend random Fourier feature maps to group-invariant kernels. Other work [21] gives explicit feature maps for additive homogeneous kernels and derives finite approximation based on spectral analysis.

Our work proposed in this paper also seeks to define explicit kernel maps but for deep multiple kernels. It is related to a recently proposed Convolutional Kernel Network [22], which

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

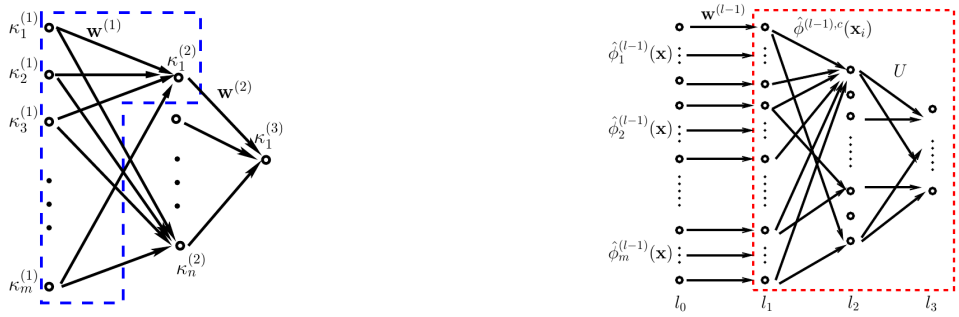

Fig. 1. The three-layer deep kernel network (in left) and the sub-module of deep map network (in right). The blue dashed area of left figure shows a sub-module of deep kernel network and each node corresponds to a kernel. In the right figure, the input is the approximated kernel map corresponding to the input kernel in the left figure and each node corresponds to a feature.

approximates convolutional Gaussian kernels through deep networks. However, our work differs from these approximation methods; indeed, we obtain deep map networks starting from *any* deep kernel network, so our proposed method is not specific to one type of kernels, but for any nonlinear and deep combination of kernels.

In this paper, we propose a novel kernel map design algorithm that transforms a given deep kernel network (DKN) into its equivalent deep map network (DMN). The method is greedy and proceed layer-wise by approaching (initial or intermediate) kernels in DKN with inner products involving explicit maps. The resulting DMN is as effective as its DKN counterpart while being highly efficient; indeed at least an order of magnitude is saved when evaluating and plugging DMN into SVMs for training and classification as shown later in this paper which is organized as follows: first, we briefly remind deep kernel networks in Section 2 and then we introduce how to build their equivalent deep map networks in Section 3. The experimental results using the ImageCLEF2013 annotation dataset are shown in Section 4, followed by the conclusion.

2. OVERVIEW OF DEEP KERNEL NETWORKS

A deep kernel network [16, 17] is a multi-layer architecture that defines a nonlinear combination of elementary kernels. Fig. 1, left shows an example of a three-layer deep kernel network. Each unit corresponds to a kernel type, i.e., elementary kernel (including Gaussian and linear in the input layer) and combination of multiple kernels for intermediate and output layers. A kernel $\kappa_p^{(l)}(\cdot,\cdot)$ for unit p at layer l is defined using a nonlinear activation function[1] g applied to a linear combination of kernels from layer $(l-1)$ (as shown in the blue dashed area in Fig. 1, left): $\{\kappa_p^{(l)}(\cdot,\cdot) = g\big(\sum_q \mathbf{w}_{p,q}^{(l-1)}\ \kappa_q^{(l-1)}(\cdot,\cdot)\big)\}$, here $\{\kappa_q^{(l-1)}(\cdot,\cdot)\}$ correspond to kernels in layer $(l-1)$, and $\{\mathbf{w}_{p,q}^{(l-1)}\}$ are the associated (learned) weights. Recursive forward feed is performed until the output layer. In the intermediate layers, hyperbolic functions are also used in order to make learning numerically more stable while at the same time ensuring that the obtained kernels are positive semi-definite.

Supervised DKN design aims to learn a discriminative kernel for classification tasks [16, 17]. Briefly speaking, gradient descent is applied to learn the network [23]. An SVM is optimized on top of DKN using hinge loss and L_2 regularization. The minimization of this regularized hinge loss is transformed into a dual maximization problem, which involves kernels. The gradient of the resulting optimization problem w.r.t the deep kernels is evaluated and used to update the underlying weights. A semi-supervised setting for deep kernel network design is also considered using an adjacency graph that model similarity between labeled and unlabeled data. Extra details about the definition and the optimization of our deep kernel network can be found in [17].

3. DEEP MAP NETWORKS

In this section, we present our main contribution: how to generate a DMN that best approximates a given DKN. The proposed method proceeds layer-wise and generates explicit maps (for different layers and units) that approximate the underlying kernels in the DKN. The resulting DMN when combined with SVMs, has *equivalent (high) performances compared to DKN, with an extra advantage of being highly efficient* as shown later in experiments.

Provided that all the elementary (input) kernels in the DKN are positive semi-definite (p.s.d) and resulting from the closure of p.s.d w.r.t different operations involved in the design of DKN[2], all kernels $\{\kappa_p^{(l)}\}_{l,p}$ will also be p.s.d. Therefore, each $\kappa_p^{(l)}(\mathbf{x},\mathbf{x}')$ can be written as $\langle \phi_p^l(\mathbf{x}), \phi_p^l(\mathbf{x}')\rangle$ with $\phi_p^l : \mathcal{X} \rightarrow \mathcal{H}$ being a mapping that takes $\mathbf{x}$ from

[1]For instance, exponential function [16].

[2]i.e., sum, product, exponential and hyperbolic activation functions (see Section 2).

the input space $\mathcal{X}$ to a high dimensional space $\mathcal{H}$. As ϕ_p^l is not necessarily explicit (known), our goal is to design an alternative (explicit) mapping $\hat{\phi}_p^l$ that guarantees $\kappa_p^{(l)}(\mathbf{x}, \mathbf{x}') \simeq \langle \hat{\phi}_p^l(\mathbf{x}), \hat{\phi}_p^l(\mathbf{x}') \rangle$. When considering these mappings through different layers, the resulting DMN generates explicit (deep) kernel features from input (shallow) features.

3.1. Input layer maps

In order to fully benefit from the explicit form of the DMN (as designed in Section 3.2), initial kernel maps should be explicitly known, in the input layer. It is clear that explicit maps are straightforward for the linear kernel and can also be defined for the polynomial. However, other kernels (more powerful and more discriminating), such as the Gaussian and the histogram intersection (HI), cannot be easily expressed with explicit maps. In this section, we will show how to define these maps for different input kernels.

Exact polynomial kernel map. the underlying kernel defined as $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^n$ can be rewritten as $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x} \otimes^n \mathbf{x}, \mathbf{x}' \otimes^n \mathbf{x}' \rangle$, with $\otimes^n$ being the Kronecker tensor product applied n times. Hence, the setting of the initial maps is $\phi_p^{(1)}(\mathbf{x}) = \mathbf{x} \otimes^n \mathbf{x}$.

Approximate HI kernel map. HI kernel is defined as $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{s} \min(\mathbf{x}^d, \mathbf{x}'^d)$ (with $\mathbf{x}^d$ being the d^{th} dimension of $\mathbf{x}$). Considering $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^s)^\intercal \in \mathcal{X}$, each dimension $\mathbf{x}^d$ of $\mathbf{x}$ is mapped using

$$\psi(\mathbf{x}^d) = 2^0 + 2^1 + \cdots + 2^{k(\mathbf{x}^d)}, \tag{1}$$

with $k(\mathbf{x}^d) = \left\lfloor Q \dfrac{\mathbf{x}^d - \ell_d}{u_d - \ell_d} \right\rfloor$; here $\lfloor z \rfloor$ stands for the largest integer not greater than $z \in \mathbb{R}$, $Q \in \mathbb{N}^+$, $\ell_d = \min_{\mathbf{x}}\{\mathbf{x}^d : \mathbf{x} \in \mathcal{X}\}$ and $u_d = \max_{\mathbf{x}}\{\mathbf{x}^d : \mathbf{x} \in \mathcal{X}\}$. Note that $\psi(.)$ is a "decimal-to-unary" map; for instance 1 is mapped to 1, 2 is mapped to 11, 3 is mapped to 111, and so on. In what follows, we reformat $\psi(\mathbf{x}^d)$ as a Q dimensional vector with its $k(\mathbf{x}^d)$ first dimensions equal to 1 and the remaining $Q - k(\mathbf{x}^d)$ dimensions equal to 0 [24].

Proposition 1 *Given any $\mathbf{x}$, $\mathbf{x}'$ in $\mathcal{X}$, for sufficiently large Q, the inner product defined as $\langle \hat{\phi}_p^{(1)}(\mathbf{x}), \hat{\phi}_p^{(1)}(\mathbf{x}') \rangle$, with*

$$\hat{\phi}_p^{(1)}(\mathbf{x}) = \left(\psi(\mathbf{x}^1)^\intercal \sqrt{\frac{u_1 - \ell_1}{Q}}, \sqrt{\ell_1}, \ldots, \psi(\mathbf{x}^s)^\intercal \sqrt{\frac{u_s - \ell_s}{Q}}, \sqrt{\ell_s} \right)^\intercal \tag{2}$$

approximates the histogram intersection kernel $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}')$.

$\psi(\mathbf{x}^d)^\intercal$ is the transpose of $\psi(\mathbf{x}^d)$. Due to limited space, the proof is omitted in this paper and can be found in [24].

Approximate Gaussian kernel map. In contrast to the previous kernels, approximate explicit maps associated to the Gaussian are obtained using eigen decomposition (ED) as a particular case of Eqs. (4), (5), i.e., with $l = 1$ (see Section 3.2).

3.2. Intermediate/output layer maps

The explicit kernel map generation is achieved layer-wise using a greedy process. Assuming all the input kernel maps obtained, the goal is to infer the maps of the subsequent layers. Recall that the intermediate/output kernels correspond to the response of nonlinear activation functions applied to linear combination of preceding (input or intermediate layer) kernels. In this subsection, we consider that the weights of these linear combinations (i.e. $\{\mathbf{w}_{p,q}^{(l)}\}$) are known; resulting from DKN learning (see Section 2).

Let $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^N$ be a training set of N samples in $\mathcal{X}$, and let $\mathbf{K}_p^l$ be the $N \times N$ gram matrix evaluated on $\mathcal{S}$ using the intermediate kernel $\kappa_p^{(l)}(\mathbf{x}, \mathbf{x}') = g\big(\langle \hat{\phi}_p^{(l-1),c}(\mathbf{x}), \hat{\phi}_p^{(l-1),c}(\mathbf{x}') \rangle\big)$; here $\hat{\phi}_p^{(l-1),c}$ stands for the concatenation of the maps at layer $l-1$, defined as

$$\hat{\phi}_p^{(l-1),c}(\mathbf{x}) = \left(\sqrt{\mathbf{w}_{p,1}^{(l-1)}} \hat{\phi}_1^{(l-1)}(\mathbf{x})^\intercal, \cdots, \sqrt{\mathbf{w}_{p,n_{l-1}}{}^{(l-1)}} \hat{\phi}_{n_{l-1}}^{(l-1)}(\mathbf{x})^\intercal \right)^\intercal, \tag{3}$$

and n_{l-1} is the number of units in layer $l-1$. For any sample $\mathbf{x}$, the projected kernel map is defined as

$$\hat{\phi}_p^{(l)}(\mathbf{x})^\intercal = \big(\kappa_p^{(l)}(\mathbf{x}, \mathbf{x}_1), \ldots, \kappa_p^{(l)}(\mathbf{x}, \mathbf{x}_N)\big)\mathbf{U}, \tag{4}$$

with $\mathbf{U} = \boldsymbol{\alpha}\boldsymbol{\Lambda}^{-1/2}$ and $\boldsymbol{\alpha}$ being the matrix of eigenvectors obtained by solving

$$\mathbf{K}_p^l \boldsymbol{\alpha} = \boldsymbol{\alpha}\boldsymbol{\Lambda}, \tag{5}$$

here $\boldsymbol{\Lambda}$ is the underlying (diagonal) matrix of eigenvalues, sorted in a descending order and columns $\{\boldsymbol{\alpha}_{.i}\}_i$ correspond to the orthonormal eigenvectors associated to the eigenvalues $\{\boldsymbol{\Lambda}_{ii}\}$. With the explicit map $\hat{\phi}_p^{(l)}$ (in Eq. 4), one may show that $\forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{S}$, $\langle \hat{\phi}_p^{(l)}(\mathbf{x}_i), \hat{\phi}_p^{(l)}(\mathbf{x}_j) \rangle = \kappa_p^{(l)}(\mathbf{x}_i, \mathbf{x}_j)$. For any samples $\mathbf{x}$, $\mathbf{x}'$ out of $\mathcal{S}$ (but drawn from the same distribution as in $\mathcal{S}$), it is clear that $\langle \hat{\phi}_p^{(l)}(\mathbf{x}), \hat{\phi}_p^{(l)}(\mathbf{x}') \rangle \rightsquigarrow \kappa_p^{(l)}(\mathbf{x}, \mathbf{x}')$ as N and the number of dimensions used in $\mathbf{U}$ increase.

3.3. Network design

Considering $\hat{\phi}_p^{(l-1)}(\mathbf{x})$ at layer $l-1$, and the maps defined in (3),(4), three layers are incrementally added in the DMN to obtain $\hat{\phi}_p^{(l)}$. The first layer provides the products between the weights $\{\mathbf{w}_{p,q}^{(l-1)}\}$ and the map $\hat{\phi}_p^{(l-1)}(\mathbf{x})$ resulting into Eq 3. The result of the first layer is fed to the second one to obtain $\{g\big(\langle \hat{\phi}_p^{(l-1),c}(\mathbf{x}), \hat{\phi}_p^{(l-1),c}(\mathbf{x}_i) \rangle\big)\}_i$; note that the weights associated to this layer correspond to $\{\hat{\phi}_p^{(l-1),c}(\mathbf{x}_i)\}_{i=1}^N$. Finally, the output of the second layer are multiplied by the weights $\mathbf{U}$ resulting into the explicit map in Eq. 4. Fig. 1, left shows an example with a sub-module (in blue dashed box) and its underlying map network in Fig. 1, right. Similar construction is achieved for all the layers where the output of each layer is fed as an input to its subsequent layer[3].

[3] As our purpose is to construct approximate deep kernel maps for a given (fixed) deep kernel network, weights $\mathbf{w}$ for the concatenation of kernel maps

4. EXPERIMENTS

In this section, we compare the performance of the proposed deep map network w.r.t deep kernel network in three aspects: i) the discrimination power, ii) the relative approximation error of DMN w.r.t DKN and iii) their efficiency. We apply both of them to image annotation task using the ImageCLEF2013 Photo Annotation database [25]. The latter has three parts: training, dev and test sets; we use only the dev set as the ground truth is available (released) for this subset only. The dev set includes 1000 images belonging to 95 concepts. In our experiments, we split the dev set into two subsets, one is used to train our deep kernels while the remaining subset is used for testing. Given a test image, the goal is to predict whether a given concept exists in that image. For that purpose, we train a "one-versus-all" SVM classifier for each concept (on top of the learned output kernel), and we use this SVM in order to predict the presence of that concept in the test image. The performances of our method are measured using the F-scores (harmonic means of recall and precision) at the concept and the sample levels (resp. MF-C and MF-S) as well as the Mean Average Precision (MAP) [25].

In our experiments, we consider a combination of 10 visual features (provided by the ImageCLEF2013 challenge) and 4 elementary kernels (including linear, polynomial with 2 orders, Gaussian[4] and histogram intersection) in order to train a three-layer deep kernel network (resulting into 40 input units and 80 hidden units, as detailed in [17]). The only difference w.r.t [17] resides in the hyperbolic activation function which is used instead of the exponential function; again, this provides a better numerical stability and convergence when optimizing the weights of the DKN using gradient descent.
Assuming the weights $\{\mathbf{w}_{p,q}^{(l-1)}\}$ of DKN known (learned), we build its equivalent DMN as shown earlier in Section 3; note that we use the whole dev set (i.e., 1000 images) for ED and we also use PCA to reduce the dimensionality of the input kernel maps (whose dimension is high). We select the eigenvectors corresponding to the largest eigenvalues that preserve 99% of the statistical variance. Table 1 shows the resulting map dimensionality obtained in the DMN (layer-by-layer).

framework	1st layer	2nd layer	3rd layer
DMN	26101	71291	766

Table 1. Map dimensionality (for the early 3 layers) in the DMN.

Given the DMN, we train SVM classifiers on top of the kernel maps associated to the training fold and we evaluate the performances on the test fold. We compare these performances w.r.t SVM classifiers trained on top of DKN. According to Table 2 we observe that both networks have similar (comparable) performances. Further comparisons using the relative error (RE) measure

$$\text{RE} = 100\times|\langle\hat{\phi}_1^{(3)},\hat{\phi}_1^{(3)}\rangle-\kappa_1^{(3)}|/(|\langle\hat{\phi}_1^{(3)},\hat{\phi}_1^{(3)}\rangle|+|\kappa_1^{(3)}|), \quad (6)$$

between the output kernels of the two networks, show that both networks produce similar gram matrices (see Table 3).
Finally, we measure the computational gain obtained with DMN against DKN. We observe, in Table 3, a noticeable gain in efficiency as DMN is (at least) an order of magnitude faster compared to its DKN counterpart; more precisely, the complexity of evaluating DMN increases linearly w.r.t the size of the database while for DKN it increases quadratically. This clearly corroborates the fact that DMN is as effective as DKN while being at the same time highly efficient and more convenient for large scale databases.

framework	MF-S	MF-C	MAP
DKN	46.23	30.00	55.73
DMN	45.84	31.54	55.16

Table 2. The performance for DKN and DMN.

database size	DMN time	DKN time	RE
1K	105	92	0.07%
2K	191	589	0.95%
3K	296	1250	1.27%
4K	405	2302	1.44%
5K	488	3761	1.54%
6K	570	5340	1.16%
7K	688	7276	1.66%
8K	757	11023	1.70%
9K	866	12779	1.73%
10K	957	18116	1.75%

Table 3. Processing time (in sec) and relative errors of DMN w.r.t DKN for different dataset cardinalities.

5. CONCLUSION

We introduced in this paper a novel method that approximates deep kernel networks with explicit maps. Our method is greedy and proceeds layer-wise by expressing (input, intermediate or output) p.s.d kernels as inner products involving explicit maps. These maps are exactly defined for some input kernels (including the linear and the polynomial) and tightly approximated for others. The strength of the method is also demonstrated through extensive experiments showing clearly the high efficiency of DMNs and their effectiveness in order to handle image annotation on the challenging ImageCLEF benchmark.
As a future work, we are currently investigating the extension of the deep map networks to other and larger scale benchmarks.

from different intermediate kernels remain fixed. However, they can also be learned using back propagation algorithms, but this is out of the main scope of this paper.

[4]with a scale parameter set to be average Euclidean distance between data samples and their immediate neighbors.

6. REFERENCES

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

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

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

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

[5] X. Qi and Y. Han, "Incorporating multiple svms for automatic image annotation," *IEEE Transcations on Knowledge and Data Engineering*, vol. 40, 2007.

[6] V. Vapnik, "Statistical learning theory," 1998.

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

[8] 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.

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

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

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

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

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

[14] A. Rakotomamonjy, F. Bach, Canu S., and Grandvalet Yves, "Simplemkl," *JMLR*, vol. 9, pp. 2491–2521, 2008.

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

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

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

[18] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *JMLR*, vol. 9, pp. 1871–1874, 2008.

[19] A. Rahimi and Recht B., "Random features for large-scale kernel machines," in *NIPS*, 2007.

[20] F. Li, Ionescu C., and Sminchisescu C., "Random fourier approximations for skewed multiplicative histogram kernels," in *DAGM conference Pattern Recognition*, 2010.

[21] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Transactions on PAMI*, vol. 34, 2012.

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

[23] 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.

[24] H. Sahbi, "Imageclef annotation with explicit context-aware kernel maps," *International Journal of Multimedia Information Retrieval*, vol. 4, pp. 113–128, 2015.

[25] M. Villegas, R. Paredes, and Thomee B., "Overview of the imageclef 2013 scalable concept image annotation subtask," in *CLEF 2013 Evaluation Labs and Workshop*, 2013.