Deep representation design from deep kernel networks

Mingyuan Jiu, Hichem Sahbi

Deep Representation Design from Deep Kernel Networks

Mingyuan Jiu[a,*], Hichem Sahbi[b]

[a]*School of Information Engineering, Zhengzhou University, Zhengzhou, China*
[b]*CNRS, LIP6 UPMC, Sorbonne University, Paris, France*

Abstract

Deep kernel learning aims at designing nonlinear combinations of multiple standard elementary kernels by training deep networks. This scheme has proven to be effective, but intractable when handling large-scale datasets especially when the depth of the trained networks increases; indeed, the complexity of evaluating these networks scales quadratically w.r.t. the size of training data and linearly w.r.t. the depth of the trained networks.

In this paper, we address the issue of efficient computation in Deep Kernel Networks (DKNs) by designing effective maps in the underlying Reproducing Kernel Hilbert Spaces (RKHS). Given a pretrained DKN, our method builds its associated Deep Map Network (DMN) whose inner product approximates the original network while being far more efficient. The design principle of our method is greedy and achieved layer-wise, by finding maps that approximate DKNs at different (input, intermediate and output) layers. This design also considers an extra fine-tuning step based on unsupervised learning, that further enhances the generalization ability of the trained DMNs. When plugged into SVMs, these DMNs turn out to be as accurate as the underlying DKNs while being at least an order of magnitude faster on large-scale datasets, as shown through extensive experiments on the challenging ImageCLEF, COREL5k benchmarks and the Banana dataset.

*Corresponding author
Email addresses: iemyjiu@zzu.edu.cn (Mingyuan Jiu), hichem.sahbi@lip6.fr (Hichem Sahbi)

Keywords: Multiple kernel learning, kernel design, deep networks, efficient computation, image annotation.

1. Introduction

Kernel design has been an active field of machine learning during the last two decades with many innovative kernel-based algorithms successfully applied to various tasks, including support vector machines (SVMs) for pattern classification and support vector regression for multivariate estimation [1, 2, 3, 4] as well as kernel-PCA for dimensionality reduction [5]. The success of these kernel-based algorithms is highly dependent on the choice of kernels; the latter are defined as symmetric and positive semi-definite functions that return similarity between data [6, 7]. Various kernels have been introduced in the literature [7] including standard elementary kernels (linear, polynomial, Gaussian, histogram intersection, etc.) as well as sophistical ones that model more complex relationships between data [3]. However, in practice, knowing a priori which (elementary or sophisticated) kernel is suitable for a given task is not obvious and research has recently been undertaken in order to train suitable kernels for different classification tasks (see for instance [8, 9, 10, 11, 12]).

Among existing solutions, Multiple Kernel Learning (MKL) [8, 13, 14] has been popular; its principle consists in learning (sparse or convex) linear combinations of elementary kernels that maximize performances for a given classification task. Different MKL algorithms have been proposed in the literature, including constrained quadratic programming [8], second-order cone and semi-infinite linear programming [13, 15] as well as "simpleMKL" based on mixed-norm regularization [14]. Wang et al. [16] also propose an alternative algorithm with hybrid kernel alignment maximization to obtain the multiple kernel coefficients. In spite of their relative success these solutions hit two major limitations: on the one hand, the convexity of these simple linear MKL models may limit the space of possible (and also relevant) solutions. On the other hand, MKL solutions, relying on shallow kernel combinations, are less powerful (compared to their deep

variants) in order to capture different levels of abstractions in the learned kernel similarity. Considering these two issues, nonlinear and deep architectures have been recently proposed and turned out to be more effective: for instance, hierarchical multiple kernel learning is proposed in [17] where elementary kernels are embedded into acyclic directed graphs while in [18], nonlinear combination of polynomial kernels are used. Following the spirit of deep convolutional neural networks [19, 20, 21], authors in [9] adopt kernel functions as a prior knowledge for regularization. In [22], Cho and Saul propose Arc-cosine kernels that mimic the computation of large neural nets which can be used in shallow as well as deep networks. In [23], a multi-layer nonlinear MKL framework is proposed, but it is restricted to only two layers; in this solution, an exponential activation function is applied to each intermediate and output kernel combination. In [24], Jiu and Sahbi extend this method to a deeper network of more than two layers, referred to as Deep Kernel Network (DKN), using a semi-supervised setting that takes into account the topology of training and test data. In all the aforementioned MKL algorithms, the computational complexity of kernel (gram-matrix) evaluation is a major issue that limits the applicability of these methods; indeed, considering a dataset with N samples, this complexity reaches $O(LN^2)$ with L being the depth of the deep kernel networks; this evaluation process is clearly intractable even on reasonable size datasets. Following the proposed DKNs in [24], we introduce explicit Deep Map Network (DMN) representations which substantially improve efficiency while maintaining high discrimination power of the original DKNs. The design principle of these DMNs is not only supervised (as in [24]), but also unsupervised and this provides better approximation accuracy.

Existing solutions that reduce the computational complexity of evaluating kernels consider explicit maps. In this respect, different solutions have been proposed in the literature including: the Nyström expansion [25] which generates low-rank kernel map approximations of original gram-matrices from uniformly

sampled data without replacement[1], and the random Fourier sampling (proposed by Rahimi an Recht [28] and extended to group-invariant kernel method in [29]) which builds explicit features for stationary kernels using random sampling of the Fourier spectrum. Nyström-based approximation is also studied in [30] for kernel subspace learning and employed for nonlinear (kernelized) update of the Ho-Kashyap algorithm using squared misclassification losses [31]. In [32], explicit feature maps for additive homogeneous kernels are given and finite approximations are derived based on spectral analysis, while in [33] data-independent random projections are studied for homogeneous polynomial kernels. Other works have been undertaken including random features [34] and convolutional kernel networks [35], which approximate maps of Gaussians using convolutional neural networks.

In this paper, we propose a novel method that reduces the computational complexity of DKN evaluation (and therefore SVM learning) on large datasets. We address the issue of kernel map approximation for *any* deep nonlinear combination of elementary kernels rather than one specific type of kernels as achieved in the aforementioned related work. Our solution relies on the *positive semi-definiteness* (p.s.d) of existing elementary kernels (linear, polynomial, etc.) and the *closure properties* of p.s.d with respect to different operations (including product, addition and exponentiation) in order to express DKN with DMN. In these closure properties, linear combinations of kernels correspond to concatenations of their respective maps, while products correspond to Kronecker tensor operations, etc. As some elementary kernels[2] used to feed the inputs of DKNs may have infinite dimensional or undefined maps, we consider new explicit maps that accurately approximate these elementary kernels. Considering these maps as inputs, this greedy process continues layer-wise in order to find all the maps of the subsequent (intermediate and output) layers. Note that the contribution presented in this paper is an extension of our preliminary work

[1] Bounds, on Nyström approximation and sampling, are given in [26, 27]

[2] such as Gaussian and Histogram Intersection.

in [36], but it differs at least in three aspects: first, we consider an unsupervised training criterion that benefits from abundant unlabeled data in order to further decrease the approximation error of the trained DMN and thereby making its generalization power as high as the underlying DKN (and also better than existing elementary and shallow kernel combinations; as shown through experiments). Furthermore, with DMNs, one may employ efficient SVM learning algorithms based on stochastic gradient descent [37] on large-scale datasets, rather than usual training algorithms that rely on heavy gram-matrices and intractable quadratic programming problems. All these statements are corroborated through extensive experiments measuring approximation accuracy and discrimination power as well as efficiency using different image annotation benchmarks (namely ImageCLEF Photo Annotation [38] and COREL5k [39]) as well as another classification task using the Banana dataset.

The rest of this paper is organized as follows: in Section 2 we first briefly remind DKNs, and then in Section 3 we introduce a novel method that builds their equivalent DMNs. In Section 4, we describe an unsupervised setting of our DMN design while in Section 5, we present the experimental validation of our method on image annotation tasks using ImageCLEF and COREL5k benchmarks. Finally, we conclude the paper while providing possible extensions for a future work.

2. Deep kernel networks at a glance

A deep kernel network [23, 24] is a multi-layered architecture that recursively defines nonlinear combinations of elementary kernels (linear, Gaussian, etc.). Let $\kappa_p^{(l)}(\cdot,\cdot)$ denote a kernel function assigned to unit p and layer l; $\kappa_p^{(l)}$ is recursively defined as the output of a nonlinear activation function[3] (denoted g) applied to a weighted combination of (input or intermediate) kernels from the

[3]For instance, exponential function [23].

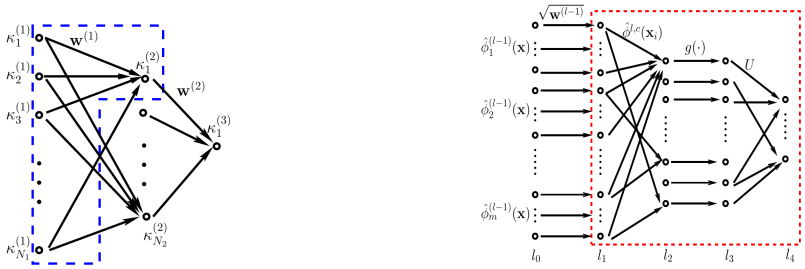

Figure 1: Left: a three-layer deep kernel network (DKN). Right: a sub-module of deep map network (DMN). The blue dash in the left figure denotes a sub-module of DKN where each node stands for a kernel. The input in the right figures are kernel maps and each unit stands for a feature.

preceding layer $(l-1)$ as

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

with $\{\mathbf{w}_{p,q}^{(l-1)}\}$ being weights connecting units at layers l and $l-1$; see the blue dashed area in Fig. 1(left). This feed-forward kernel evaluation is achieved layer-wise till reaching the final output kernel. In this recursive definition, other activation functions g can be chosen (particularly for the intermediate layers) including the hyperbolic making the learning numerically more stable while also preserving the p.s.d of the final output kernel.

For a given classification task, the weights $\{\mathbf{w}_{p,q}^{(l-1)}\}$ are trained discriminatively [23, 24] using a max margin SVM criterion which aims at minimizing a regularized hinge loss on top of the learned DKN. This results into an SVM optimization problem which is solved in its dual form by backpropagating the gradient of that form w.r.t. the output kernel using the chain rule [19]. Then, the weights connecting layers in the DKN are updated using gradient descent. Variants of this optimization criterion, leveraging both labeled and unlabeled data (following a semi-supervised and laplacian setting) makes it possible to train better DKN as detailed in [24].

3. Deep map networks

In this section, we introduce a novel method that finds for any given DKN, its associated DMN; the proposed method proceeds layer-wise by finding explicit maps that best fit the original kernels in the DKN. As shown later in experiments, this process delivers highly efficient DMNs, while being comparably accurate w.r.t. their underlying DKNs. Later in Section 4, we introduce an extension that further enhances the approximation quality of our DMN; starting from the initial weights of the DMN, we update these weights by minimizing the difference between inner products of the maps in the DMN and the original kernels in the DKN. The strength of this extension also resides in its unsupervised setting which makes it possible to learn from abundant unlabeled sets.

Considering all the elementary (input) kernels in the DKN as positive semi-definite and resulting from the closure of the p.s.d w.r.t. different operations (including sum, product, exponential and hyperbolic activation functions), all the intermediate and output kernels $\{\kappa_p^{(l)}\}_{l,p}$ will also be p.s.d. Each $\kappa_p^{(l)}(\mathbf{x}, \mathbf{x}')$ can therefore be written as an inner product of kernel maps as $\langle \phi_p^l(\mathbf{x}), \phi_p^l(\mathbf{x}') \rangle$, with $\phi_p^l : \mathcal{X} \to \mathcal{H}$ being a mapping from the input space $\mathcal{X}$ to a high dimensional space $\mathcal{H}$. As the explicit form of ϕ_p^l is not necessarily explicit (known), our goal is to design an approximated 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 these approximated mappings through different layers are known, the resulting DMN provides deep kernel representations from the input data.

3.1. Input layer maps

In order to fully benefit from DMNs, the maps of the elementary kernels, that feed these DMNs, should be explicitly known. As discussed earlier, different kernels have different maps; for linear and polynomial, their maps are straightforward and can be easily defined. However, for other more powerful and discriminating kernels, such as the Gaussian and the histogram intersection (HI), their maps are either infinite dimensional or unknown. In this subsection, the definitions of exact and approximate explicit maps are shown for different kernels (including polynomial and HI).

Exact polynomial kernel map. A polynomial kernel defined as $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^{n+1}$ can be expressed as $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x} \otimes^n \mathbf{x}, \mathbf{x}' \otimes^n \mathbf{x}' \rangle$, with $\otimes^n$ standing for the Kronecker tensor product applied n times; this tensor product on two matrices A (of size $m \times n$) and B (of size $p \times q$) results into a block matrix (of size $mp \times nq$) as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}. \tag{2}$$

Hence, it is easy to see that the exact explicit map for a polynomial kernel is $\phi_p^{(1)}(\mathbf{x}) = \mathbf{x} \otimes^n \mathbf{x}$.

Approximate HI kernel map. The approximate explicit maps of HI can be obtained using vector quantization. Given two vectors $\mathbf{x}$ and $\mathbf{x}'$ of dimension s, the HI on $\mathbf{x}$, $\mathbf{x}'$ 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 value of d^{th} dimension of $\mathbf{x}$). Considering $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^s)^\top \in \mathcal{X}$, each dimension $\mathbf{x}^d$ of $\mathbf{x}$ is mapped to

$$\psi(\mathbf{x}^d) = 2^0 + 2^1 + \cdots + 2^{k(\mathbf{x}^d)}, \tag{3}$$

where $k(\mathbf{x}^d) = \left\lfloor Q \dfrac{\mathbf{x}^d - \ell_d}{u_d - \ell_d} \right\rfloor$ and $\lfloor z \rfloor$ stands for the largest integer not greater than $z \in \mathbb{R}$, $Q \in \mathbb{N}^+$ is a predefined quantization, $\ell_d = \min_{\mathbf{x}}\{\mathbf{x}^d : \mathbf{x} \in \mathcal{X}\}$ and $u_d = \max_{\mathbf{x}}\{\mathbf{x}^d : \mathbf{x} \in \mathcal{X}\}$. In the above definition, $\psi(.)$ is a "decimal-to-unary" map; for instance, 1 is mapped to 1, 2 is mapped to 11, 3 to 111, and so on. In the following, $\psi(\mathbf{x}^d)$ is rewritten as a vector of Q dimensions, and its first $k(\mathbf{x}^d)$ dimensions are set to 1 and the remaining are set to 0 [40].

Proposition 1. *Given any $\mathbf{x}$, $\mathbf{x}'$ in $\mathcal{X}$, for sufficiently large Q, the inner product $\langle \hat{\phi}_p^{(1)}(\mathbf{x}), \hat{\phi}_p^{(1)}(\mathbf{x}') \rangle$ approximates the histogram intersection kernel $\kappa_p^{(1)}(\mathbf{x}, \mathbf{x}')$, where*

$$\hat{\phi}_p^{(1)}(\mathbf{x}) = \left(\psi(\mathbf{x}^1)^\top \sqrt{\frac{u_1 - \ell_1}{Q}}, \sqrt{\ell_1}, \dots, \psi(\mathbf{x}^s)^\top \sqrt{\frac{u_s - \ell_s}{Q}}, \sqrt{\ell_s} \right)^\top \tag{4}$$

is the approximate kernel map and $\psi(\mathbf{x}^d)^\top$ stands for the transpose of $\psi(\mathbf{x}^d)$.

Proof. The proof is given in the Appendix A. □

Approximate Gaussian kernel map. As the explicit map of the Gaussian kernel is infinite dimensional, we consider instead an approximate explicit map of that kernel using eigen decomposition (ED) as shown in Eqs. (6), (5) with $l = 1$ (see Section 3.2). This ED is not restricted to the Gaussian kernel and can also be extended to other kernels whose exact explicit maps are difficult to obtain.

3.2. Intermediate/output layer maps

Given the explicit map of each elementary kernel at the input layer, our goal is to design the maps of the subsequent layers. Since the map of each layer depends on its preceding layers, this goal is achieved layer-wise using a greedy process. As intermediate/output kernels in the DKN are defined as linear combinations of kernels in the preceding (input or intermediate) layers followed by nonlinear activations, we mainly focus on how to approximate the maps of these activation functions in the DMN; in this section, we assume that weights $\{\mathbf{w}_{p,q}^{(l)}\}$ connecting different layers are already known resulting from the initial setting of the DKN (see again Section 2).

Proposition 2. *Let $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^N$ be a subset of N samples of $\mathcal{X}$, and let $\mathbf{K}_p^l$ be a gram-matrix whose entries are defined on $\mathcal{S}$. Let $\mathbf{U}_p^{(l)} = \boldsymbol{\alpha}\boldsymbol{\Lambda}^{-1/2}$ with $\boldsymbol{\alpha}$, $\boldsymbol{\Lambda}$ being respectively the matrices of eigenvectors and eigenvalues obtained by solving*

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

Considering $\|.\|_2$ as the ℓ_2 (matrix) norm and $\hat{\mathbf{K}}_p^l$ as the gram-matrix associated to $\{\langle\hat{\phi}_p^{(l)}(\mathbf{x}), \hat{\phi}_p^{(l)}(\mathbf{x}')\rangle\}_{\mathbf{x},\mathbf{x}'\in\mathcal{S}}$ with

$$\hat{\phi}_p^{(l)}(\mathbf{x})^\top = \Big(g(\langle\hat{\phi}_p^{l,c}(\mathbf{x}), \hat{\phi}_p^{l,c}(\mathbf{x}_1)\rangle) \dots g(\langle\hat{\phi}_p^{l,c}(\mathbf{x}), \hat{\phi}_p^{l,c}(\mathbf{x}_N)\rangle)\Big)\mathbf{U}_p^{(l)} \tag{6}$$

and

$$\hat{\phi}_p^{l,c}(\mathbf{x}) = \left(\sqrt{\mathbf{w}_{p,1}^{(l-1)}}\hat{\phi}_1^{(l-1)}(\mathbf{x})^\top \cdots \sqrt{\mathbf{w}_{p,n_{l-1}}^{(l-1)}}\hat{\phi}_{n_{l-1}}^{(l-1)}(\mathbf{x})^\top\right)^\top, \tag{7}$$

then the following property is satisfied

$$\left\|\hat{\mathbf{K}}_p^l - \mathbf{K}_p^l\right\|_2 = 0. \tag{8}$$

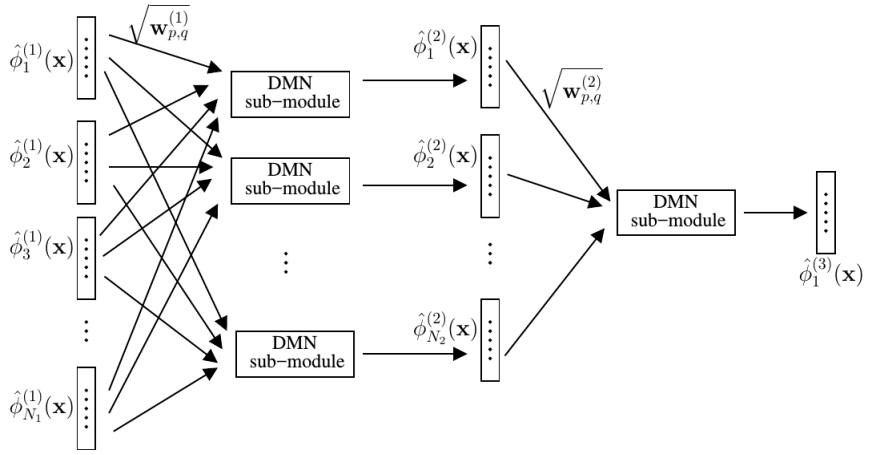

Figure 2: The flowchart of DMN for a three-layer DKN, as shown in Fig. 1(left).

Proof. The proof is given in the Appendix B. □

Note that for any samples $\mathbf{x}$, $\mathbf{x}'$ taken out of $\mathcal{S}$ (but with similar distribution as $\mathcal{S}$), it is clear (as also observed in our experiments) that $|\langle\hat{\phi}_p^{(l)}(\mathbf{x}), \hat{\phi}_p^{(l)}(\mathbf{x}')\rangle - \kappa_p^{(l)}(\mathbf{x}, \mathbf{x}')| \rightsquigarrow 0$ as N and the number of eigenvectors used in $\{\mathbf{U}_p^{(l)}\}$ increase.

3.3. Network design

We incrementally expand each layer l in the DKN into three sub-layers in the underlying DMN in order to design the map $\hat{\phi}_p^{(l)}$. The first sub-layer provides the products between weights $\{(\mathbf{w}_{p,q}^{(l-1)})^{1/2}\}_q$ and the preceding maps $\{\hat{\phi}_q^{(l-1)}(\mathbf{x})\}_q$ resulting into the intermediate map $\hat{\phi}_p^{l,c}(\mathbf{x})$ as shown in Eq. (7). Afterwards, we feed this map $\hat{\phi}_p^{l,c}(\mathbf{x})$ to Eq. (6) in two steps: (i) in the second sub-layer, inner products are achieved between $\hat{\phi}_p^{l,c}(\mathbf{x})$ and parameters $\{\hat{\phi}_p^{l,c}(\mathbf{x}_i)\}_{i=1}^N$ followed by the activations $\{g(.)\}_{i=1}^N$ (with g being the hyperbolic excepting the final layer in the DKN which uses the exponential); (ii) in the third sub-layer, the explicit map $\hat{\phi}_p^{(l)}$ is obtained as the product of $\{g(.)\}_{i=1}^N$ and weights $\mathbf{U}_p^{(l)}$. Fig. 1(right) shows a sub-module of DMN with sub-layers and Fig. 2 provides the flowchart of DMN for a three-layer DKN. Similarly, all the subsequent layers in the DMN

are designed by processing the DKN layer-wise.[4]

4. Enhancing DMN Parameters

So far the design principle of our method (shown in Section 3.3 and Fig. 1) seeks to find explicit maps whose inner products approximate the original kernel values. This is achieved by expanding each layer in the DKN into three sub-layers in the DMN with parameters fixed to $\{\hat{\phi}_p^{l,c}(\mathbf{x}_i)\}_i$ and $\mathbf{U}_p^{(l)}$. In spite of being efficient and also effective w.r.t. the DKN (see experiments), the resulting DMN can be further improved when re-training and fine-tuning these parameters as shown subsequently.

The purpose of the proposed unsupervised algorithm is to further reduce the approximation error between the kernel values from DKN and the inner product of kernel maps from DMN. Let $\mathcal{S}' \subset \mathcal{X}$ be a subset drawn from the same distribution as $\mathcal{S}$ and define $\mathcal{P}$ as a subset of pairs taken from $\mathcal{S}' \times \mathcal{S}'$. Our goal is to optimize maps of DMN using the following unsupervised criterion

$$E = \sum_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}} \frac{1}{2}\left\|\hat{\phi}_1^{(L)}(\mathbf{x})^\top \hat{\phi}_1^{(L)}(\mathbf{x}') - \kappa_1^{(L)}(\mathbf{x},\mathbf{x}')\right\|^2, \tag{9}$$

where $\kappa_1^{(L)}(\mathbf{x},\mathbf{x}')$ corresponds to the kernel value obtained using the DKN and $\hat{\phi}_1^{(L)}(\mathbf{x})$, $\hat{\phi}_1^{(L)}(\mathbf{x}')$ are the underlying (unknown) kernel maps; initially, only $\{\hat{\phi}_p^{(1)}(\mathbf{x}), \hat{\phi}_p^{(1)}(\mathbf{x}')\}_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}}$ are known according to the procedure shown in Section 3.1.

Considering the initial setting of DMN parameters (i.e., $\{\hat{\phi}_p^{l,c}(\mathbf{x}_i)\}_i$ and $\mathbf{U}_p^{(l)}$), the learning process of this DMN relies on backpropagation [19]. The latter finds the best parameters by minimizing the objective function (E) following an "end-to-end" framework where the gradients of E are given using the chain rule; we firstly compute the gradients of the loss function E w.r.t. final kernel maps,

[4] As the goal, in this paper, is to build approximate deep kernel maps for a given (fixed) deep kernel network, the weights $\mathbf{w}$ between different layers remain fixed (as shown in Eq. (7)). However, they can also be jointly learned using gradient descent, but this is out of the main scope of this paper.

then we backpropagate them through the DMN in order to obtain the gradients w.r.t. the parameters of DMN, finally we average them over training pairs to obtain the descent direction and update DMN parameters.

Starting from the derivative of E w.r.t. $\hat{\phi}_1^{(L)}(\mathbf{x})$

$$\frac{\partial E}{\hat{\phi}_1^{(L)}(\mathbf{x})} = \big(\hat{\phi}_1^{(L)}(\mathbf{x})^\top \hat{\phi}_1^{(L)}(\mathbf{x}') - \kappa_1^{(L)}(\mathbf{x},\mathbf{x}')\big)\hat{\phi}_1^{(L)}(\mathbf{x}'), \tag{10}$$

we obtain the gradients w.r.t. different layers $l = L, \ldots, 1$ and units $p = 1, \ldots, n_l$. As the construction of DMN is achieved layer-wise (see again Section 3.3), we show below the backpropagation procedure for a module (shown in Fig. 1, right). Given the derivatives of E w.r.t. $\hat{\phi}_p^{(l)}(\mathbf{x})$ in layer l, we evaluate the derivatives w.r.t. $\hat{\phi}_q^{(l-1)}(\mathbf{x})$ in layer $(l-1)$. The derivative w.r.t. $\hat{\phi}_p^{(l)}(\mathbf{x})$ is backpropagated to $\kappa_p^{(l)}$ in Eq. (6) by

$$\frac{\partial E}{\partial \kappa_p^{(l)}(\mathbf{x},\mathbf{x}_i)} = (\frac{\partial E}{\partial \hat{\phi}_p^{(l)}(\mathbf{x})})^\top [\mathbf{U}_p^{(l)}]_i^\top, \tag{11}$$

here $[.]_i$ stands for the i-th row of a matrix. Considering $\kappa_p^{(l)}(\mathbf{x},\mathbf{x}_i) = g\big(f_p^{(l)}(\mathbf{x},\mathbf{x}_i)\big)$, with $f_p^{(l)}(\mathbf{x},\mathbf{x}_i) = \langle \hat{\phi}_p^{l,c}(\mathbf{x}), \hat{\phi}_p^{l,c}(\mathbf{x}_i)\rangle$, we obtain

$$\frac{\partial E}{\partial f_p^{(l)}(\mathbf{x},\mathbf{x}_i)} = g'(f_p^{(l)}(\mathbf{x},\mathbf{x}_i))\ \frac{\partial E}{\partial \kappa_p^{(l)}(\mathbf{x},\mathbf{x}_i)}, \tag{12}$$

where $g'(\cdot)$ is the derivative of the nonlinear activation function; for instance, $g'(\cdot) = 1 - \tanh(\cdot)^2$ for the tangent hyperbolic and $g'(\cdot) = g(\cdot)$ for the exponential. By accumulating the derivatives from each term $f_p^{(l)}(\mathbf{x},\mathbf{x}_i)$, we obtain

$$\frac{\partial E}{\hat{\phi}_p^{l,c}(\mathbf{x})} = \sum_{i=1}^{N} \hat{\phi}_p^{l,c}(\mathbf{x}_i) \frac{\partial E}{\partial f_p^{(l)}(\mathbf{x},\mathbf{x}_i)}, \tag{13}$$

Finally, we get the derivatives w.r.t. $\hat{\phi}_q^{(l-1)}(\mathbf{x})$ for layer $(l-1)$ in Eq. (7) by

$$\frac{\partial E}{\partial \hat{\phi}_q^{(l-1)}(\mathbf{x})} = \sqrt{\mathbf{w}_{p,q}^{(l-1)}}\mathrm{Frag}(\frac{\partial E}{\hat{\phi}_p^{l,c}(\mathbf{x})})_q, \tag{14}$$

where $\mathrm{Frag}(\frac{\partial E}{\hat{\phi}_p^{l,c}(\mathbf{x})})_q$ stands for the fragment of derivatives corresponding to the kernel maps of the unit q at layer $(l-1)$ in the DKN.

The gradients of the loss function E w.r.t. $\mathbf{U}_p^{(l)}$ and $\hat{\phi}_p^{l,c}(\mathbf{x}_i)$ are then given as

$$\Delta\mathbf{U}_p^{(l)} = (\kappa_p^{(l)}(\mathbf{x},\mathbf{x}_1)\dots\kappa_p^{(l)}(\mathbf{x},\mathbf{x}_N))^\top \ (\frac{\partial E}{\partial\hat{\phi}_p^{(l)}(\mathbf{x})})^\top \tag{15}$$

$$\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i) = \frac{\partial E}{\partial f_p^{(l)}(\mathbf{x},\mathbf{x}_i)}\hat{\phi}_p^{l,c}(\mathbf{x}). \tag{16}$$

Error backpropagation is achieved layer-wise from the final to the input layer; the increments of $\{\hat{\phi}_p^{l,c}(\mathbf{x}_i)\}_{i=1}^N$ and $\mathbf{U}_p^{(l)}$ are obtained by Eq. (16) and Eq. (15). Gradient descent with a step η (see experiments) is performed to update the parameters of DMN. The whole learning procedure is shown in Algorithm 1.

As described earlier, an initial DMN is firstly set using the training set $\mathcal{S}$, then sample pairs in $\mathcal{P}$ are randomly selected from $\mathcal{S}'$ to further enhance the parameters of the new (fine-tuned) DMN. As a result, the fine-tuned DMN enables us to obtain a better approximation of the original DKN on large datasets while being highly efficient as shown through the following experiments in image annotation.

5. Experiments

In this section, we compare the performance of the proposed DMN w.r.t. its underlying DKN in three aspects: i) discrimination power, ii) relative approximation error between DMN and DKN and iii) also efficiency. The targeted task is image annotation; given a picture, the goal is to predict a list of keywords that best describes the visual content of that image. We consider two challenging and widely used annotation benchmarks: ImageCLEF [38], COREL5k [39], and also Banana (see details below). For three sets, we learn – highly competitive – 3-layer DKNs using the setting in [24] and we plug these DKNs into SVMs in order to achieve classification and annotation.

The discrimination power of the learned DMN and DKN networks is measured following the protocol defined by challenge organizers and data providers (see [38] for ImageCLEF and [39] for COREL5k; see also extra details below).

Input: Fixed $\{\mathbf{w}_{p,q}^{(l-1)}\}$ $(l = 2, \ldots, L)$,

A set of sample pairs $\mathcal{P}$,

Kernel maps $\{\hat{\phi}_p^{(1)}(\mathbf{x})\}_{\mathbf{x}\in\mathcal{S}'}$ at the input layer,

Output kernel values $\{\kappa_1^{(L)}(\mathbf{x}, \mathbf{x}')\}_{(\mathbf{x},\mathbf{x}')\in\mathcal{P}}$.

Initialization: $\{\hat{\phi}_p^{l,c}(\mathbf{x}_i)\}_{i=1}^N$ and $\{\mathbf{U}_p^{(l)}\}$, $p = \{1, \ldots, n_l\}$, learning rate η.

Output: Optimal (updated) $\{\hat{\phi}_p^{l,c}(\mathbf{x}_i)\}_{i=1}^N$ and $\{\mathbf{U}_p^{(l)}\}$.

repeat

  for *each pair* $(\mathbf{x}, \mathbf{x}') \in \mathcal{P}$ **do**

    Forward $(\hat{\phi}_p^{(1)}(\mathbf{x}), \hat{\phi}_p^{(1)}(\mathbf{x}'))$ through DMN to obtain

    $(\hat{\phi}_1^{(L)}(\mathbf{x}), \hat{\phi}_1^{(L)}(\mathbf{x}'))$by Eqs. (7), (6);

    Compute the loss by Eq. (9);

    Compute the gradients $\frac{\partial E}{\hat{\phi}_1^{(L)}(\mathbf{x})}$ by Eq. (10);

    for $l = L : 2$ **do**

      Backward the gradients $\frac{\partial E}{\hat{\phi}_1^{(L)}(\mathbf{x})}$ by Eqs. (11)-(14);

      Compute $(\Delta U_p^{(l)})_{\mathbf{x}}$ and $(\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i))_{\mathbf{x}}$ by Eq. (15) and (16);

      Compute the gradients from $\hat{\phi}_p^{(L)}(\mathbf{x}')$: $(\Delta U_p^{(l)})_{\mathbf{x}'}$ and

      $(\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i))_{\mathbf{x}'}$;

      Average both gradients: $\Delta U_p^{(l)} \leftarrow \frac{1}{2}\big((\Delta U_p^{(l)})_{\mathbf{x}} + (\Delta U_p^{(l)})_{\mathbf{x}'}\big)$;

      $\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i) \leftarrow \frac{1}{2}\big((\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i))_{\mathbf{x}} + (\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i))_{\mathbf{x}'}\big)$;

      Update these parameters by gradient descents;

      $U_p^{(l)} \leftarrow U_p^{(l)} - \eta\Delta U_p^{(l)}$;

      $\hat{\phi}_p^{l,c}(\mathbf{x}_i) \leftarrow \hat{\phi}_p^{l,c}(\mathbf{x}_i) - \eta\Delta\hat{\phi}_p^{l,c}(\mathbf{x}_i)$;

    end

  end

until *Convergence*;

Algorithm 1: Unsupervised DMN learning algorithm

The relative approximation error (RE) of a given DMN w.r.t. its underlying DKN is measured (on a given set $\mathcal{T} \subset \mathcal{X}$) as

$$\mathrm{RE} = \frac{1}{|\mathcal{T}|^2} \sum_{\mathbf{x},\mathbf{x}' \in \mathcal{T}} \frac{|\langle \hat{\phi}_1^{(3)}(\mathbf{x}), \hat{\phi}_1^{(3)}(\mathbf{x}') \rangle - \kappa_1^{(3)}(\mathbf{x},\mathbf{x}')|}{|\langle \hat{\phi}_1^{(3)}(\mathbf{x}), \hat{\phi}_1^{(3)}(\mathbf{x}') \rangle| + |\kappa_1^{(3)}(\mathbf{x},\mathbf{x}')|} \times 100\%, \tag{17}$$

In the remainder of this section, we show different evaluation measures (discrimination power, RE and efficiency) on ImageCLEF and COREL5k benchmarks; note that efficiency was measured on a Mac OS with Intel Core i5 processors.

5.1. ImageCLEF benchmark

The ImageCLEF Photo Annotation benchmark [38] includes more than 250k (training, dev and test) images belonging to 95 different concepts. As ground truth is available (released) only on the dev set (with 1,000 images), we learn DKNs and SVMs [24] using only the dev set; the latter is split into two subsets: the first one used for DKN+SVM training while the other one for SVM testing. Given a concept and a test image, the decision about whether that concept is present in that test image depends on the score of a classifier; the latter corresponds to a "one-versus-all" SVM that returns a positive score if the concept is present in the test image and a negative score otherwise. We employ the LIBSVM library [41] in order to train each SVM independently, and we obtain the optimal trade-off parameter C_k ($\in [2^{-10}, \ldots, 2^{10}]$) using 3-fold cross-validation on the training set. The discrimination power of DKN and DMN (when combined with SVMs) is evaluated using the F-measure (defined as harmonic means of recalls and precisions) both at the concept and the image levels (resp. denoted MF-C and MF-S) as well as the Mean Average Precision (MAP) [38]; high values of these measures imply better performances.

In order to feed the inputs of DKN, we consider a combination of 10 visual features (provided by the ImageCLEF challenge organizers) and 4 elementary kernels (i.e. linear, polynomial with 2 orders, Gaussian[5] and histogram intersection) and we train a three-layer DKN with 40 input and 80 hidden units in

[5] with a scale hyper-parameter set to be average Euclidean distance between data samples and their neighbors.

a supervised way following the scheme in [24]; the only difference w.r.t. [24] resides in the hyperbolic tangent activation function which is used to provide a better numerical stability and convergence when training DKN.

Initial DMNs. Assuming the weights $\{\mathbf{w}_{p,q}^{(l-1)}\}$ of three-layer DKN known, we build its equivalent DMN (referred to as initial DMN) as shown in Section 3. In these experiments, we consider two random samplings of the subset $\mathcal{S}$ – from the dev set with $|\mathcal{S}| = 500$ and $|\mathcal{S}| = 1000$ – in order to build the initial DMN (see Section 3 and Eqs. (7), (6)). According to Table 1, we observe that the performance of the initial DMN – with $|\mathcal{S}| = 500$ – slightly degrades compared to its underlying DKN; indeed, MF-S and MF-C decrease by 1.3 and 2.6 pts respectively while MAP decreases by 6.0 pts. With $|\mathcal{S}| = 1000$ performances of the initial DMN is clearly improved compared to the one with $|\mathcal{S}| = 500$; we obtain a slight gain in MF-S and comparable performance in MF-C. We also provide a comparison of the discrimination power of inital DMN against shallow DKN (i.e two-layer DKN) using a supervised setting; Table 1 clearly shows the superiority of initial DMN (when $|\mathcal{S}| = 1000$). The relative approximation error (RE) of the two initial DMNs (i.e., with $|\mathcal{S}| = 500$ and $|\mathcal{S}| = 1000$) are also shown in Table 2; we evaluate these REs on $\mathcal{T}$ with a cardinality ranging from 2,000 to 10,000 samples. From these results, we observe that REs are comparably low on small sets; indeed, with $|\mathcal{T}| = 2,000$, the obtained REs are equal to 0.94% when $|\mathcal{S}| = 500$ and 0.95% when $|\mathcal{S}| = 1000$. Higher REs are obtained on larger $\mathcal{T}$ and this clearly motivates the importance of fine-tuning in order to make REs (and thereby performances) of the learned DMN stable (and close to the underlying DKN).

Fine-tuned DMNs. In order to fine-tune the parameters of DMN, we use the learning procedure presented in Section 4. We consider an unlabeled set $\mathcal{S}'$ (with $|\mathcal{S}'|$ ranging from 1,000 to 4,000) and we randomly sample 100,000 pairs from $\mathcal{S}' \times \mathcal{S}'$ to minimize criterion (9) using gradient descent with a step-size empirically set to 10^{-6}, a mini-batch size equal to 200 and a max number of iterations set to 5,000. Fig. 3 shows the evolution of the approximation loss (9) as the learning process iterates; we clearly observe that 100,000 pairs

Framework	MF-S	MF-C	MAP
2-layer DKN	44.96	25.77	53.95
3-layer DKN	46.23	30.00	55.73
Initial DMN ($\|\mathcal{S}\| = 500$)	44.92	27.39	49.75
Fine-tuned DMN ($\|\mathcal{S}'\| = 2000$)	**45.05**	**27.51**	**49.80**
Fine-tuned DMN ($\|\mathcal{S}'\| = 3000$)	44.94	27.40	49.80
Fine-tuned DMN ($\|\mathcal{S}'\| = 4000$)	45.06	27.44	49.79
Initial DMN ($\|\mathcal{S}\| = 1000$)	47.73	29.40	53.15
Fine-tuned DMN ($\|\mathcal{S}'\| = 2000$)	47.79	29.68	52.89
Fine-tuned DMN ($\|\mathcal{S}'\| = 3000$)	**47.95**	**29.80**	**53.32**
Fine-tuned DMN ($\|\mathcal{S}'\| = 4000$)	47.70	29.30	53.33

Table 1: The discrimination power (in %) of different DMNs w.r.t. the underlying DKN; in these experiments, two initial DMNs are designed using 500 and 1000 samples.

Configuration	$\|\mathcal{S}'\|$	2K	3K	4K	5K	6K	7K	8K	9K	10K
Initial DMN ($\|\mathcal{S}\| = 500$)	-	0.94	1.25	1.41	1.51	1.58	1.62	1.66	1.69	1.71
Fine-tuned DMN	500	0.89	1.19	1.35	1.45	1.52	1.57	1.60	1.63	1.65
	1000	0.89	1.20	1.36	1.46	1.53	1.58	1.61	1.64	1.66
	2000	0.42	0.46	0.50	0.52	0.54	0.56	0.57	0.58	0.59
	3000	0.52	0.47	0.47	0.47	0.47	0.47	0.48	0.48	0.48
	4000	0.60	0.51	0.49	0.47	0.47	0.46	0.46	0.46	0.46
Initial DMN ($\|\mathcal{S}\| = 1000$)	-	0.95	1.27	1.44	1.54	1.62	1.67	1.70	1.74	1.76
Fine-tuned DMN	1000	0.89	1.21	1.38	1.48	1.55	1.60	1.64	1.67	1.69
	2000	0.37	0.41	0.44	0.46	0.48	0.49	0.50	0.51	0.52
	3000	0.46	0.43	0.43	0.43	0.44	0.44	0.44	0.45	0.45
	4000	0.54	0.48	0.46	0.45	0.45	0.44	0.44	0.44	0.44

Table 2: Relative errors of initial and fine-tuned DMNs w.r.t. the DKN for different dataset cardinalities $|\mathcal{T}|$ (ranging from 2K to 10K) and when two different initializations are employed.

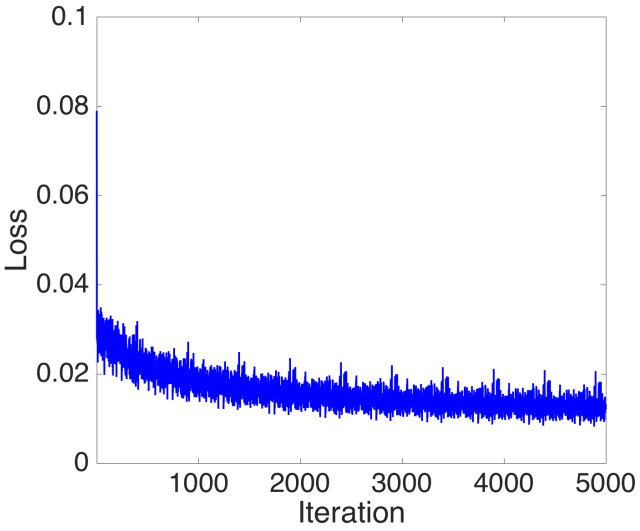

Figure 3: This figure shows the loss criterion in Eq. (9) as the learning iterates when $|\mathcal{S}| = 500$ and $|\mathcal{S}'| = 2000$.

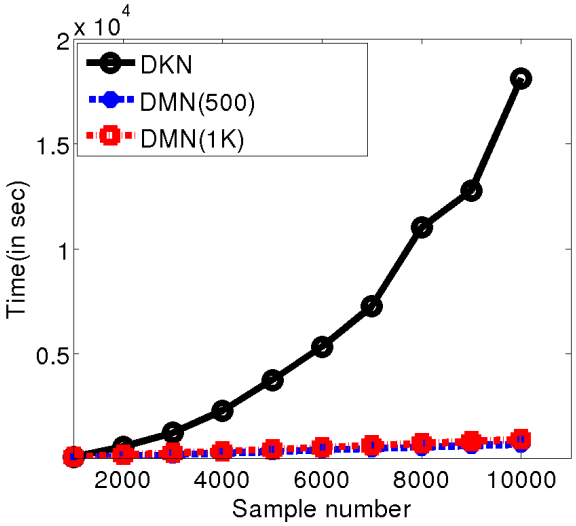

Figure 4: This figure shows a comparison of processing time between two different DMNs and their underlying DKN as $|\mathcal{T}|$ increases (with $|\mathcal{S}| = 500$ and $|\mathcal{S}| = 1000$) on ImageCLEF dataset.

	$\|\mathcal{T}\|$	50K	100K
3-layer DKN	Time	40.4 hrs	160.3 hrs
Fine-tuned DMN	Time	1.1 hrs	2.4 hrs
$\|\mathcal{S}\| = 500$, $\|\mathcal{S}'\| = 4000$	RE	0.46%	0.46%
Fine-tuned DMN	Time	1.3 hrs	2.8 hrs
$\|\mathcal{S}\| = 1000$, $\|\mathcal{S}'\| = 4000$	RE	0.45%	0.45%

Table 3: This table shows a comparison of processing time and relative errors between the DKN and the fine-tuned DMN on $50K$ and $100K$ images of ImageCLEF."hrs" stands for "hours".

are already sufficient to train accurate DMNs. With an extra fine tuning step (shown subsequently), the accuracy of these DMNs is further improved.

As shown in Table 1, we observe that the discrimination power of different DMNs remains stable (with a slight gain in MF-S when $|\mathcal{S}'| = 3000$) w.r.t. their underlying DKNs, and this naturally follows the noticeably small REs of the fine-tuned DMNs (see Table 2). The latter are further positively impacted when $|\mathcal{S}'|$ becomes larger; for instance, when increasing $|\mathcal{S}'|$ from 1,000 to 4,000, the RE decreases significantly (particularly when $|\mathcal{T}| = 10,000$). Moreover, and in contrast to the initial DMNs, the fine-tuned DMNs are less sensitive to $|\mathcal{T}|$ as shown through the observed REs which remain stable w.r.t. $|\mathcal{T}|$.

Finally, we measure the gain in efficiency obtained with DMNs against DKNs. From Fig. 4, we observe that DMN is (at least) an order of magnitude faster compared to its DKN; for instance, with 10,000 samples, DKN requires more than 15,000 seconds in order to compute kernel values while DMN requires less than 1,000 seconds. Table 3 also provides a comparison of efficiency and RE on much larger sets (resp. 50K and 100K) randomly sampled from the (unlabeled) training set of ImageCLEF; a significant improvement in efficiency is observed. In other words, the complexity of evaluating DMNs is linear while for DKN it is quadratic. These results clearly corroborate the fact that the proposed DMNs are as effective as DKNs while being highly efficient especially

on large scale datasets.

5.2. COREL5k benchmark

The COREL5k database introduced in [39] is another benchmark which is widely used for image annotation. In this database, 4,999 images are collected and a vocabulary of 200 keywords is used for annotation. This set is split into two parts; the first one includes 4,500 images for training and the second one 499 images for testing. As for ImageCLEF, the task is again to assign a list of keywords for each image in the test set.

Each image in COREL5k is described using 15 types of INRIA features [42] including: GIST features, 6 color histograms for RGB, HSV, LAB in two spatial layouts, 8 bag-of-features based on SIFT and robust hue descriptors in two spatial layouts. Following the standard protocol defined on COREL5k [39], each test image is annotated with up to 5 keywords and performances (discrimination power of image classification/annotation) are measured by the mean precision and recall over keywords (referred to as $\mathbf{P}$ and $\mathbf{R}$ respectively) as well as the number of keywords with non-zero recall value (denoted $\mathbf{N}_+$); again, higher values of these measures imply better performances.

As in ImageCLEF (see section 5.1), we use 4 elementary kernels for each feature: linear, order two polynomial, RBF (with a scale parameter set to the average distance between data) and histogram intersection; in total, we use 60 different elementary kernels as inputs to the 3-layer DKN. We also use the same DKN architecture on COREL5k with a slight difference in the number of units in the hidden layers (equal to 120 instead of 80 in ImageCLEF). Again, the weights of DKN are learned using the semi-supervised learning procedure presented in [24] where the similarity between images $\mathbf{x}$ and $\mathbf{x}'$ is computed by the heat kernel $S(\mathbf{x}, \mathbf{x}') = \exp^{\frac{-||\mathbf{x}-\mathbf{x}'||^2}{4t}}$, where t is the mean distance between data samples and their neighbors. An ensemble of "one-versus-all" SVM classifiers is trained on top of DKN for each category using LIBSVM [41] and the trade-off parameter C_k is also chosen by 3-fold cross-validation on the training set. The average decision score from all the classifiers is taken as a final score

Framework	**R**	**P**	$\mathbf{N}_+$
3-layer DKN	37.65	25.49	158
Initial DMN ($\|\mathcal{S}\| = 500$)	31.30	18.67	155
Fine-tuned DMN ($\|\mathcal{S}'\| = 2000$)	31.34	18.54	155
Fine-tuned DMN ($\|\mathcal{S}'\| = 3000$)	31.62	18.43	153
Fine-tuned DMN ($\|\mathcal{S}'\| = 4000$)	31.18	19.04	155
Fine-tuned DMN ($\|\mathcal{S}'\| = 4999$)	**31.65**	**19.13**	**157**
Initial DMN ($\|\mathcal{S}\| = 700$)	32.31	19.39	155
Fine-tuned DMN ($\|\mathcal{S}'\| = 2000$)	32.57	19.82	157
Fine-tuned DMN ($\|\mathcal{S}'\| = 3000$)	33.05	**20.88**	**159**
Fine-tuned DMN ($\|\mathcal{S}'\| = 4000$)	33.08	20.40	158
Fine-tuned DMN ($\|\mathcal{S}'\| = 4999$)	**33.30**	20.18	158

Table 4: The discrimination power of different DMNs w.r.t. the underlying DKN on COREL5k; in these experiments, two initial DMNs are designed using 500 and 700 samples.

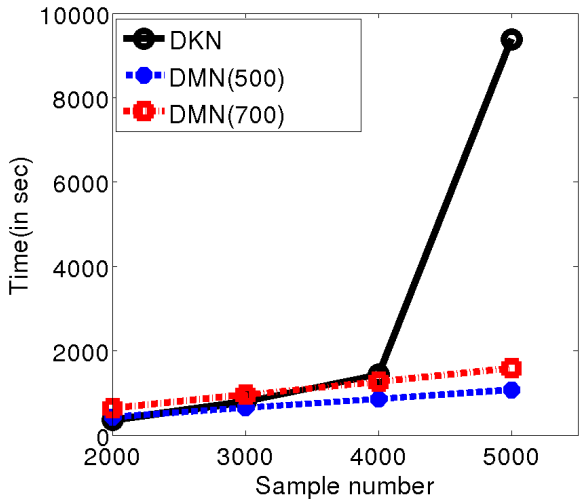

Figure 5: Comparison of processing time between two approximated DMNs (with $|\mathcal{S}| = 500$ and $|\mathcal{S}| = 700$) and their underlying DKN as $|\mathcal{T}|$ increases on COREL5k dataset.

Framework	$\lvert\mathcal{S}'\rvert$	2K	3K	4K	4999
Initial DMN $\lvert\mathcal{S}\rvert = 500$	-	2.45	2.41	2.35	2.26
Fine-tuned DMN	500	1.22	1.28	1.32	1.37
	1000	1.23	1.35	1.40	1.42
	2000	1.12	1.15	1.19	1.19
	3000	1.14	1.12	1.13	1.12
	4000	1.18	1.14	1.11	1.10
	4999	1.18	1.14	1.12	1.10
Initial DMN $\lvert\mathcal{S}\rvert = 700$	-	2.43	2.39	2.33	2.24
Fine-tuned DMN	700	1.30	1.42	1.48	1.51
	1000	1.22	1.35	1.42	1.44
	2000	1.09	1.13	1.17	1.18
	3000	1.11	1.10	1.11	1.11
	4000	1.16	1.12	1.09	1.08
	4999	1.16	1.12	1.10	1.08

Table 5: Relative errors of initial and fine-tuned DMNs (w.r.t. the underlying DKN) on COREL5k as $\lvert\mathcal{T}\rvert$ increases (with values ranging from 2K to 4999)

Method	Learned Input feat.	context	**R**	**P**	$\mathbf{N}_+$
CRM [43]	no	no	19	16	107
InfNet [44]	no	no	24	17	112
JEC-15 [45]	no	yes	33	28	140
TagPop σML [42]	no	yes	42	33	160
wTKML [46]	no	yes	42	21	173
LDMKL [47]	no	yes	44	29	179
CNN-R [48]	yes	yes	41.3	32.0	166
3-layer DKN+SVM [49]	no	no	37.7	25.5	158
Init. DMN+SVM ($\|\mathcal{S}\| = 700$)	no	no	32.3	19.3	155
FT DMN+SVM ($\|\mathcal{S}\| = 700$)	no	no	33.1	20.9	159
Init. DMN+SVM ($\|\mathcal{S}\| = 1200$)	no	no	34.0	20.9	162
FT DMN+SVM ($\|\mathcal{S}\| = 1200$)	no	no	34.7	21.0	168
ResNet[50] + SVM	yes	no	34.5	21.8	161
3-layer DKN+SVM [49]	yes	no	42.6	24.9	180
Init. DMN+SVM ($\|\mathcal{S}\| = 700$)	yes	no	36.1	21.7	166
FT DMN+SVM ($\|\mathcal{S}\| = 700$)	yes	no	36.8	22.4	165
Init. DMN+SVM ($\|\mathcal{S}\| = 1000$)	yes	no	37.4	21.6	162
FT DMN+SVM ($\|\mathcal{S}\| = 1000$)	yes	no	37.7	22.3	164
Init. DMN+SVM ($\|\mathcal{S}\| = 1200$)	yes	no	37.8	23.2	167
FT DMN+SVM ($\|\mathcal{S}\| = 1200$)	yes	no	38.9	23.2	169

Table 6: Extra comparison of the proposed DMN w.r.t different settings as well as the related work. In these experiments, $|\mathcal{S}'| = 3000$ and different $|\mathcal{S}|$ are used. In this table, FT stands for Fine-Tuned.

for a given category. In order to avoid the severe imbalanced class distributions in SVM training, we adopt a sampling strategy that randomly selects a subset of negative samples whose cardinality is equal to the number of positive training samples. Hence, each classifier is learned using all the positive data and a random subset of negative data. The discrimination power of the learned DKNs+SVMs is shown in Table 4.

Initial and fine-tuned DMNs. Assuming the weights $\{\mathbf{w}_{p,q}^{(l-1)}\}$ of DKN known (learned), we build the initial DMN as shown in Section 3. We consider two random samplings of the subset $\mathcal{S}$ – from the training set with $|\mathcal{S}| = 500$ and $|\mathcal{S}| = 700$ – in order to build the initial DMN. We also use the learning procedure presented in Section 4 in order to fine-tune the parameter of the DMN. We consider an unlabeled set $\mathcal{S}'$ which includes up to 4,999 samples (i.e. the whole COREL5k set); again we randomly sample 100,000 pairs in order to minimize the criterion in Eq. (9) using gradient descent with a step-size empirically set to 10^{-6}, a mini-batch size equal to 200 and a max number of iterations set to 5,000.

According to Table 4, we observe that the performances of the initial DMNs ($\mathbf{R}$, $\mathbf{P}$ and $\mathbf{N}_+$) again degrade compared to their underlying DKNs as a result of the high RE of these DMNs. This degradation in performances is also amplified by the scarceness of training data for SVM learning in COREL5k (in contrast to ImageCLEF) especially when the RE is relatively large (see Table. 5). However, the discrimination power is improved when more data are used to design these DMNs (i.e., with $|\mathcal{S}| = 700$ and also $|\mathcal{S}| = 1200$ in Table 6). Furthermore, fine-tuning DMNs reduces the RE as $|\mathcal{S}'|$ increases, and makes RE stable even with a relatively large $|\mathcal{T}|$, so RE (on COREL5k) behaves similarly compared to ImageCLEF. Finally, Fig. 5 shows a comparison of processing time between DMN and DKN. It is easy to see that when $|\mathcal{T}|$ is small, the processing times of DKN and DMN are comparable. However, when $|\mathcal{T}|$ reaches large values (e.g., $|\mathcal{T}| = 4999$), DMN becomes an order of magnitude faster than its underlying DKN while maintaining a comparable accuracy.

Extra comparisons. We further compare the performance of DMNs against

two generative methods (i.e., CRM [43] and infNet [44]) and several discriminative methods (i.e., JEC-15 [45], TagPop σML [42], wTKML [46], LDMKL [47], CNN-R [48]) for image annotation. CRM [43] and infNet [44] learn optimal joint probability distributions between features and semantic labels while JEC-15 [45] and TagPop σML [42] (based on KNN) define classification criteria for images by weighting labels of their neighbors. wTKML [46] and LDMKL [47] are the most closely related (kernel) methods; wTKML learns explicit and transductive kernel maps using a priori knowledge taken from the statistical (semantic and geometric) dependencies between classes while LDMKL combines Laplacian SVM with deep kernel networks using an "end-to-end" learning framework. CNN-R [48] adopts convolutional neural networks for annotation, that combines deep features from Caffe-Net with word2vec embedding. As introduced in the literature, these related methods leverage different sources of contexts and a priori knowledge while our method does not.

In our experiments (see Table 6), we use four elementary kernels (linear, polynomial, RBF and HI) combined with different features as inputs to the designed DKN and DMN networks: "handcrafted features" including GIST and SIFT and "learned features" taken from ResNet [50] (pretrained on the ImageNet) which is a very deep architecture consisting of 152 layers; the 2048 dimensional features of the last pooling layer are used in our annotation task. Using all these elementary kernels and features, we first train a DKN in a supervised way according to [49], then we design and fine-tune its associated DMNs with $|\mathcal{S}| = 700$ and $|\mathcal{S}'| = 3000$ (as done in Table. 4).

From the results shown in Table 6, first, we observe that the use of ResNet features as inputs to our DMN framework provides a clear gain compared to the use of handcrafted features. Second, fine-tuning DMNs bring a clear gain compared to the initial DMNs as well as ResNet. Our DKN (and its DMN variant) can even catch (and sometimes outperform) the aforementioned related work which again relies on different contextual clues, in contrast to our method. We believe that considering context will further enhance the performance of DKNs and their associated DMNs, but this is out of the main scope of this paper and

will be investigated as a future work.

Finally, Fig. 6 shows examples of annotation results, on the test set, obtained using the learned DMNs and the underlying DKNs on ImageCLEF and COREL5k datasets. From these figures, DMNs behave similarly, w.r.t. DKNs, with an extra advantage of being computationally more efficient especially on COREL5k (as shown in Table 7); whereas the computational complexity of DKN evaluation scales linearly w.r.t. the number of support vectors (which is an order of magnitude larger on COREL5k w.r.t. ImageCLEF: 4,500 versus 500), the computational complexity of DMN evaluation grows slowly and remains globally stable w.r.t. the number of support vectors (which is again an order of magnitude larger on COREL5k). These results are also consistent with those already shown in Fig. 4 and Fig. 5.

Dataset	Framework	time (in sec)
ImageCLEF	DKN	0.68
	Fine-tuned DMN ($\|\mathcal{S}\| = 500$)	0.57
	Fine-tuned DMN ($\|\mathcal{S}\| = 1000$)	0.95
COREL5k	DKN	10.39
	Fine-tuned DMN ($\|\mathcal{S}\| = 500$)	1.22
	Fine-tuned DMN ($\|\mathcal{S}\| = 700$)	1.58
	Fine-tuned DMN ($\|\mathcal{S}\| = 1000$)	2.51
	Fine-tuned DMN ($\|\mathcal{S}\| = 1200$)	3.67

Table 7: Comparison of the average processing time per test image (excluding feature extraction) on ImageCLEF and COREL5k datasets.

5.3. Banana dataset

In Sections 5.1 and 5.2, we studied the efficiency and the effectiveness of the proposed method in image annotation. In this section, we further investigate the applicability of our method to another classification problem using the Banana dataset [51]. The latter differs from ImageCLEF and COREL5k

Figure 6: Examples of annotation results using DKNs and their "Fine-tuned" DMN variants on ImageCLEF (top) and COREL5k (bottom)."GT" stands for ground-truth keywords and the symbol "*" stands for the presence of a keyword in a given test image.

Method	LDKL [51]	SDMKL [49]	Initial DMNs	Fine-tuned DMNs
Accuracy	88.67	91.07	90.21	91.05

Table 8: This table shows the comparison in accuracy of LDKL, kernel-based semi-supervised learning of 3-layer deep kernel network (SDMKL), inital DMNs and fine-tuned DMNs on Banana dataset.

in that (i) it corresponds to vectors of measurements[6] rather than images, and (ii) the labels associated to these measurements are exclusive. The Banana dataset corresponds to a binary classification problem, which contains 1,000 samples for training and 4,300 for testing. Following the standard experimental protocol [51], we first evaluate 21 RBF kernels (with scale factors ranging from $2^{-10}M$ to $2^{10}M$; here M is the average Euclidean distance between samples and their neighbors), then we train a three-layer deep kernel network (referred

[6]Similar to the spirit of industrial scenarios where data are collected using specific instruments.

to as SDMKL [49]) using a semi-supervised setting. In these experiments, we set the SVM trade-off parameter C_k to 0.25 using cross validation on a random subset containing 20% of training data. We also consider another baseline for comparison referred to as LDKL (Local Deep Kernel Learning) [51]. The accuracy of LDKL and SDMKL as well as our DMNs (designed and fine-tuned as described in Sections 3, 4) are shown in Tab. 8. From this table, we observe that the performances of the initial DMNs are close to their original DKNs and the fine-tuned DMNs further improve these performances, and this clearly validates the applicability of our method to different classification problems.

6. Conclusion

In this paper we introduce a novel method that transforms deep kernel networks (DKNs) into highly efficient deep map networks (DMNs). DKNs, as nonlinear and multi-layered combinations of standard kernels, are highly effective but computationally very demanding especially when handling large-scale problems. In order to reduce the computational complexity of DKNs, the proposed method defines a DMN architecture layer-wise by expressing positive semi-definite kernels in different (input, intermediate, and output) layers of DKN as inner products involving explicit maps. As also theoritically shown, these maps are either exactly designed for input kernels (including linear and polynomial) or tightly approximated (for intermediate and output kernels in DKN) with at least an order of magnitude gain both in kernel and SVM evaluation while maintaining a comparable classification accuracy. An extra fine-tuning step makes it possible to further enhance the accuracy of DMNs; this step, totally unsupervised, benefits from large unlabeled sets in order to further minimize the difference between the inner products of the designed maps and the original DKNs in the associated Reproducing Kernel Hilbert Space. Extensive experiments on the challenging ImageCLEF and COREL5k benchmarks for image annotation as well as the Banana dataset, clearly demonstrate the effectiveness of DMNs and their high efficiency. As a future work, we are currently

studying the memory burden of DMNs that may arise as a function of their (high) dimensionality and depth; this issue will be tackled by coupling DMNs with auto-encoders.

Appendix A: Proof of Proposition 1

Proof. For pair of samples $(\mathbf{x}_i, \mathbf{x}_j), \forall i, j \in \{1, \ldots, m\}$, we have:

$$\begin{aligned} \langle \hat{\phi}_p^{(1)}(\mathbf{x}_i), \hat{\phi}_p^{(1)}(\mathbf{x}_j) \rangle &= \langle \psi(\mathbf{x}_i^1), \psi(\mathbf{x}_j^1) \rangle (\tfrac{u_1 - \ell_1}{Q}) + \ell_1 + \ldots \\ &+ \langle \psi(\mathbf{x}_i^s), \psi(\mathbf{x}_j^s) \rangle (\tfrac{u_s - \ell_s}{Q}) + \ell_s \end{aligned} \tag{18}$$

It is easy to see that $\forall d \in \{1, \ldots, s\}$, $\langle \psi(\mathbf{x}_i^d), \psi(\mathbf{x}_j^d) \rangle = \min(k(\mathbf{x}_i^d), k(\mathbf{x}_j^d))$. By replacing in Eq. (18)

$$\begin{aligned} & \left| \langle \hat{\phi}_p^{(1)}(\mathbf{x}_i), \hat{\phi}_p^{(1)}(\mathbf{x}_j) \rangle - \kappa_p^{(1)}(\mathbf{x}_i, \mathbf{x}_j) \right| \\ = & \left| \sum_{d=1}^{s} \min(k(\mathbf{x}_i^d), k(\mathbf{x}_j^d)) \frac{u_d - \ell_d}{Q} + \ell_d - \min(\mathbf{x}_i^d, \mathbf{x}_j^d) \right| \\ \leq & \sum_{d=1}^{s} \left| \left\lfloor Q \frac{\min(\mathbf{x}_i^d, \mathbf{x}_j^d) - \ell_d}{u_d - \ell_d} \right\rfloor \frac{u_d - \ell_d}{Q} + \ell_d - \min(\mathbf{x}_i^d, \mathbf{x}_j^d) \right| \\ = & \sum_{d=1}^{s} \frac{u_d - \ell_d}{Q} \left| \left\lfloor Q \frac{\min(\mathbf{x}_i^d, \mathbf{x}_j^d) - \ell_d}{u_d - \ell_d} \right\rfloor - Q \frac{\min(\mathbf{x}_i^d, \mathbf{x}_j^d) - \ell_d}{u_d - \ell_d} \right| \\ \leq & \frac{1}{Q} \sum_{d=1}^{s} u_d - \ell_d, \quad (\text{as } \; |\lfloor z \rfloor - z| \leq 1), \end{aligned} \tag{19}$$

as Q increases, $\left| \langle \hat{\phi}_p^{(1)}(\mathbf{x}_i), \hat{\phi}_p^{(1)}(\mathbf{x}_j) \rangle - \kappa_p^{(1)}(\mathbf{x}_i, \mathbf{x}_j) \right| \rightsquigarrow 0.$ □

Appendix B: Proof of Proposition 2

Proof. Let's proceed layer-wise by induction; for $l = 1$ (and following Section 3.1), the initial kernel maps $\{\hat{\phi}_p^{(1)}(.)\}$ are designed to satisfy $\hat{\phi}_p^{(1)}(\mathbf{x})^\top \hat{\phi}_p^{(1)}(\mathbf{x}') = \kappa_p^{(1)}(\mathbf{x}, \mathbf{x}')$.

Now provided that $\hat{\phi}_q^{(l-1)}(\mathbf{x})^\top \hat{\phi}_q^{(l-1)}(\mathbf{x}') = \kappa_q^{(l-1)}(\mathbf{x}, \mathbf{x}')$, the property to show is $\hat{\phi}_p^{(l)}(\mathbf{x})^\top \hat{\phi}_p^{(l)}(\mathbf{x}') = \kappa_p^{(l)}(\mathbf{x}, \mathbf{x}')$, $\forall \mathbf{x} \in \mathcal{S}$. Following (7) we have

$$\begin{aligned} \langle \hat{\phi}_p^{l,c}(\mathbf{x}), \hat{\phi}_p^{l,c}(\mathbf{x}') \rangle &= \sum_{q=1}^{n_{l-1}} \mathbf{w}_{p,q}^{(l-1)} \; \hat{\phi}_q^{(l-1)}(\mathbf{x})^\top \hat{\phi}_q^{(l-1)}(\mathbf{x}') \\ &= \sum_{q=1}^{n_{l-1}} \mathbf{w}_{p,q}^{(l-1)} \; \kappa_q^{(l-1)}(\mathbf{x}, \mathbf{x}'), \end{aligned} \tag{20}$$

the second equality results from the hypothesis of induction. By plugging (20) into (6), we obtain

$$\hat{\phi}_p^{(l)}(\mathbf{x})^\top = \big(\kappa_p^{(l)}(\mathbf{x},\mathbf{x}_1),\ldots,\kappa_p^{(l)}(\mathbf{x},\mathbf{x}_N)\big)\mathbf{U}_p^{(l)}, \tag{21}$$

and equivalently $\hat{\mathbf{K}}_p^l = \mathbf{K}_p^l\ \mathbf{U}_p^{(l)}\ \mathbf{U}_p^{(l)\top}\ \mathbf{K}_p^l$. Hence,

$$\begin{aligned} \left\|\hat{\mathbf{K}}_p^l - \mathbf{K}_p^l\right\|_2 &= \left\|\mathbf{K}_p^l\ \boldsymbol{\alpha}\boldsymbol{\Lambda}^{-1/2}\ \boldsymbol{\Lambda}^{-1/2}\ \boldsymbol{\alpha}^\top\ \mathbf{K}_p^l - \mathbf{K}_p^l\right\|_2 \\ &= \left\|\boldsymbol{\alpha}\boldsymbol{\Lambda}\ \boldsymbol{\Lambda}^{-1}\ \boldsymbol{\alpha}^\top\ \mathbf{K}_p^l - \mathbf{K}_p^l\right\|_2 \\ &= \left\|\boldsymbol{\alpha}\ \boldsymbol{\alpha}^\top\ \mathbf{K}_p^l - \mathbf{K}_p^l\right\|_2 \\ &= \left\|\mathbf{K}_p^l - \mathbf{K}_p^l\right\|_2 = 0 \end{aligned} \tag{22}$$

which also results from Eq. (5) and the orthogonality of eigenvectors in $\boldsymbol{\alpha}$. □

Acknowledgement

This work was supported by a grant from National Natural Science Foundation of China (No. 61806180) and in part by a grant from the research agency ANR (Agence Nationale de la Recherche) under the MLVIS project (ANR-11-BS02-0017).

References

References

[1] B. Caputo, C. Wallraven, M. Nilsback, Object categorization via local kernels, in: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), Vol. 2, 2004, pp. 132–135.

[2] S. Lyu, Mercer kernels for object recognition with local features, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, 2005, pp. 223–229.

[3] K. Grauman, T. Darrell, The pyramid match kernel: Efficient learning with sets of features, Journal of Machine Learning Research 8 (2007) 725–760.

[4] X. Qi, Y. Han, Incorporating multiple svms for automatic image annotation, Pattern Recognition 40 (2) (2007) 728–741.

[5] K. Q. Weinberger, F. Sha, L. K. Saul, Learning a kernel matrix for nonlinear dimensionality reduction, in: International Conference on Machine Learning (ICML), 2014, pp. 839–846.

[6] V. Vapnik, Statistical learning theory, Wiley, New York, 1998.

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

[8] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, M. I. Jordan, Learning the kernel matrix with semi-definite programming, Journal of Machine Learning Research 5 (2004) 27–72.

[9] K. Yu, W. Xu, Y. Gong, Deep learning with kernel regularization for visual recognition, in: Advances in Neural Information Processing Systems (NIPS), 2008, pp. 1889–1896.

[10] C. Corinna, M. Mehryar, R. Afshin, Two-stage learning kernel algorithms, in: International Conference on Machine Learning (ICML), 2010, pp. 239–246.

[11] H. Sahbi, J.-Y. Audibert, R. Keriven, Context-dependent kernels for object classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2011) 699–708.

[12] H. Sahbi, X. Li, Context-based support vector machines for interconnected image annotation, in: Asian Conference on Computer Vision (ACCV), 2011, pp. 214–227.

[13] F. Bach, G. Lanckriet, M. Jordan, Multiple kernel learning, conic duality, and the smo algorithm, in: International Conference on Machine Learning (ICML), 2004, pp. 1–6.

[14] A. Rakotomamonjy, F. Bach, C. S., G. Yves, Simplemkl, Journal of Machine Learning Research 9 (2008) 2491–2521.

[15] S. Sonnenburg, G. Rätsch, C. Schafer, B. Schölkopf, Large scale multiple kernel learning, Journal of Machine Learning Research 7 (2006) 1531–1565.

[16] Y. Wang, X. Liu, Y. Dou, Q. Lv, Y. Lu, Multiple kernel learning with hybrid kernel alignment maximization, Pattern Recognition 70 (2017) 104–111.

[17] F. Bach, Exploring large feature spaces with hierarchical multiple kernel learning, in: Advances in Neural Information Processing Systems (NIPS), 2009, pp. 1–9.

[18] C. Cortes, M. Mohri, A. Rostamizadeh, Learning non-linear combinations of kernels, in: Advances in Neural Information Processing Systems (NIPS), 2009, pp. 1–9.

[19] Y. LeCun, L. Botto, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of IEEE 86 (11) (1998) 2278–2324.

[20] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), Vol. 60, 2012, pp. 1097–1105.

[21] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8) (2013) 1915–1929.

[22] Y. Cho, L. Saul, Kernel methods for deep learning, in: Advances in Neural Information Processing Systems (NIPS), Vol. 28 (1), 2009, pp. 342–350.

[23] J. Zhuang, I. Tsang, S. Hoi, Two-layer multiple kernel learning, in: International Conference on Machine Learning (ICML), 2011, pp. 909–917.

[24] M. Jiu, H. Sahbi, Semi supervised deep kernel design for image annotation, in: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 1156–1160.

[25] C. Williams, M. Seeger, Using the nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems (NIPS), 2001, pp. 682–688.

[26] P. Drineas, M. Mahoney, On the nyström method for approximating a gram matrix for improved kernel-based learning, Journal of Machine Learning Research 6 (2005) 2153–2175.

[27] S. Kumar, M. Mohri, A. Talwalkar, Sampling methods for the nyström method, Journal of Machine Learning Research 13 (1) (2012) 981–1006.

[28] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems (NIPS), Vol. 20, 2007, pp. 1177–1184.

[29] F. Li, C. Ionescu, C. Sminchisescu, Random fourier approximations for skewed multiplicative histogram kernels, in: DAGM conference Pattern Recognition, 2010, pp. 262–271.

[30] A. Iosifidis, M. Gabbouj, Nyström-based approximate kernel subspace learning, Pattern Recognition 57 (2016) 190–197.

[31] C. Zhu, D. Gao, Improved multi-kernel classification machine with nyström approximation technique, Pattern Recognition 48 (2015) 1490–1509.

[32] A. Vedaldi, A. Zisserman, Efficient additive kernels via explicit feature maps, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3) (2012) 480–492.

[33] D. López-Sánchez, A. G. Arrieta, J. M. Corchado, Data-independent random projections from the feature-space of the homogeneous polynomial kernel, Pattern Recognition 82 (2018) 130–146.

[34] P. Huang, L. Deng, M. Hasegawa-Johnson, X. He, Random features for kernel deep convex network, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013, pp. 3143–3147.

[35] J. Mairal, P. Koniusz, Z. Harchaoui, C. Schmid, Convolutional kernel networks, in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 2627–2635.

[36] M. Jiu, H. Sahbi, Deep kernel map networks for image annotation, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 1571–1575.

[37] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, Liblinear: A library for large linear classification, Journal of Machine Learning Research 9 (2008) 1871–1874.

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

[39] P. Duygulu, K. Barnard, N. de Freitas, D. Forsyth, Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary, in: European Conference on Computer Vision (ECCV), 2002, pp. 97–112.

[40] H. Sahbi, Imageclef annotation with explicit context-aware kernel maps, International Journal of Multimedia Information Retrieval 4 (2015) 113–128.

[41] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 1–27.

[42] M. Guillaumin, T. Mensink, J. Verbeek, C. Schmid, Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation, in: 2009 IEEE 12th International Conference on Computer Vision (ICCV), 2009, pp. 309–316.

[43] V. Lavrenko, R. Manmatha, J. Jeon, A model for learning the semantics of pictures, in: Advances in Neural Information Processing Systems (NIPS), 2003, pp. 553–560.

[44] D. Metzler, R. Manmatha, A inference network approach to image retrieval, in: International Conference on Image and Video Retrieval (CIVR), 2004, pp. 42–50.

[45] A. Makadia, V. Pavlovic, S. Kumar, A new baseline for image annotation, in: European Conference on Computer Vision (ECCV), 2008, pp. 316–329.

[46] P. Vo, H. Sahbi, Transductive kernel map learning and its application to image annotation, in: The British Machine Vision Conference (BMVC), 2012, pp. 1–12.

[47] D. Zhang, M. Islam, G. Lu, A review on automatic image annotation techniques, Journal of the China Society for Scientific and Technical Information 45 (1) (2013) 346–362.

[48] V. N. Murthy, S. Maji, R. Manmatha, Automatic image annotation using deep learning representations, in: International Conference on Multimedia Retrieval, 2015, pp. 603–606.

[49] M. Jiu, H. Sahbi, Nonlinear deep kernel learning for image annotation, IEEE Transactions on Image Processing 26 (4) (2017) 1820–1832.

[50] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[51] C. Jose, P. Goyal, P. Aggrwal, M. Varma, Local deep kernel learning for efficient non-linear svm prediction, in: International Conference on Machine Learning (ICML), 2013, pp. 486–494.