Open access • Journal Article • DOI:10.1109/TPAMI.2013.83

Distance-Based Image Classification: Generalizing to New Classes at Near-Zero Cost — Source link

Thomas Mensink, Jakob Verbeek, Florent Perronnin, Gabriela Csurka

Institutions: University of Amsterdam, French Institute for Research in Computer Science and Automation, Xerox

Published on: 01 Nov 2013 - IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE)

Topics: Support vector machine, Classifier (UML) and Contextual image classification

Distance-Based Image Classification: Generalizing to new classes at near-zero cost

Thomas Mensink, Jakob Verbeek, Florent Perronnin, Gabriela Csurka

Distance-Based Image Classification: Generalizing to new classes at near-zero cost

Thomas Mensink, *Member IEEE,* Jakob Verbeek, *Member, IEEE,*
Florent Perronnin, and Gabriela Csurka

Abstract—We study large-scale image classification methods that can incorporate new classes and training images continuously over time at negligible cost. To this end we consider two distance-based classifiers, the k-nearest neighbor (k-NN) and nearest class mean (NCM) classifiers, and introduce a new metric learning approach for the latter. We also introduce an extension of the NCM classifier to allow for richer class representations. Experiments on the ImageNet 2010 challenge dataset, which contains over 10^6 training images of 1,000 classes, show that, surprisingly, the NCM classifier compares favorably to the more flexible k-NN classifier. Moreover, the NCM performance is comparable to that of linear SVMs which obtain current state-of-the-art performance. Experimentally we study the generalization performance to classes that were not used to learn the metrics. Using a metric learned on 1,000 classes, we show results for the ImageNet-10K dataset which contains 10,000 classes, and obtain performance that is competitive with the current state-of-the-art, while being orders of magnitude faster. Furthermore, we show how a zero-shot class prior based on the ImageNet hierarchy can improve performance when few training images are available.

Index Terms—Metric Learning, k-Nearest Neighbors Classification, Nearest Class Mean Classification, Large Scale Image Classification, Transfer Learning, Zero-Shot Learning, Image Retrieval

1 Introduction

In this paper we focus on the problem of large-scale, multi-class image classification, where the goal is to assign automatically an image to one class out of a finite set of alternatives, *e.g.* the name of the main object appearing in the image, or a general label like the scene type of the image. To ensure scalability, often linear classifiers such as linear SVMs are used [1], [2]. Additionally, to speed-up classification, dimension reduction techniques could be used [3], or a hierarchy of classifiers could be learned [4], [5]. The introduction of the ImageNet dataset [6], which contains more than 14M manually labeled images of 22K classes, has provided an important benchmark for large-scale image classification and annotation algorithms. Recently, impressive results have been reported on 10,000 or more classes [1], [3], [7]. A drawback of these methods, however, is that when images of new categories become available, new classifiers have to be trained from scratch at a relatively high computational cost.

Many real-life large-scale datasets are open-ended and dynamic: new images are continuously added to existing classes, new classes appear over time, and the semantics of existing classes might evolve too. Therefore, we are interested in distance-based classifiers which enable the addition of new classes and new images to existing classes at (near) zero cost. Such methods can be used continuously as new data becomes available, and additionally alternated from time to time with a computationally heavier method to learn a good metric using all available training data. In particular we consider two distance-based classifiers.

The first is the k-nearest neighbor (k-NN) classifier, which uses all examples to represent a class, and is a highly non-linear classifier that has shown competitive performance for image classification [3], [7], [8], [9]. New images (of new classes) are simply added to the database, and can be used for classification without further processing.

The second is the nearest class mean classifier (NCM), which represents classes by their mean feature vector of its elements, see *e.g.* [10]. Contrary to the k-NN classifier, this is an efficient linear classifier. To incorporate new images (of new classes), the relevant class means have to be adjusted or added to the set of class means. In Section 3, we introduce an extension which uses several prototypes per class, which allows a trade-off between the model complexity and the computational cost of classification.

The success of these methods critically depends on the used distance functions. Therefore, we cast our classifier learning problem as one of learning a low-rank Mahalanobis distance which is shared across all classes. The dimensionality of the low-rank matrix is used as regularizer, and to improve computational and storage efficiency.

In this paper we explore several strategies for learning such a metric. For the NCM classifier, we propose a novel metric learning algorithm based on multi-class logistic discrimination (NCMML), where a sample from a class is enforced to be closer to its class mean than to any other

- *Thomas Mensink*
  ISLA Lab - University of Amsterdam
  E-mail: firstname.lastname@uva.nl
- *Jakob Verbeek*
  LEAR Team - INRIA Grenoble
  E-mail: firstname.lastname@inria.fr
- *Florent Perronnin and Gabriela Csurka*
  Xerox Research Centre Europe
  E-mail: firstname.lastname@xrce.xerox.com

class mean in the projected space. We show qualitatively and quantitatively the advantages of our NCMML approach over the classical Fisher Discriminant Analysis [10]. For k-NN classification, we rely on the Large Margin Nearest Neighbor (LMNN) framework [11] and investigate two variations similar to the ideas presented in [11], [12] that significantly improve classification performance.

Most of our experiments are conducted on the ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC'10) dataset, which consists of 1.2M training images of 1,000 classes. To apply the proposed metric learning techniques on such a large-scale dataset, we employ stochastic gradient descend (SGD) algorithms, which access only a small fraction of the training data at each iteration [13]. To allow metric learning on high-dimensional image features of datasets that are too large to fit in memory, we use in addition product quantization [14], a data compression technique that was recently used with success for large-scale image retrieval [15] and classifier training [1].

As a baseline approach, we follow the winning entry of the ILSVRC'11 challenge [1]: Fisher vector image representations [16] are used to describe images and one-vs-rest linear SVM classifiers are learned independently for each class. Surprisingly, we find that the NCM classifier outperforms the more flexible k-NN classifier. Moreover, the NCM classifier performs on par with the SVM baseline, and shows competitive performance on new classes.

This paper extends our earlier work [17], as follows. First, for the NCM classifier, in Section 3, we compare the NCMML metric learning to the classic FDA, we introduce an extension which uses multiple centroids per class, we explore a different learning objective, and we examine the critical points of the objective. Second, in Section 4, we provide more details on the SGD triplet sampling strategy used for LMNN metric learning, and we present an efficient gradient evaluation method. Third, we extend the experimental evaluation with an experiment where NCMML is used to learn a metric for instance level image retrieval.

The rest of the paper is organized as follows. We first discuss a selection of related works which are most relevant to this paper. In Section 3 we introduce the NCM classifier and the NCMML metric learning approach. In Section 4 we review LMNN metric learning for k-NN classifiers. We present extensive experimental results in Section 5, analyzing different aspects of the proposed methods and comparing them to the current state-of-the-art in different application settings such as large scale image annotation, transfer learning and image retrieval. Finally, we present our conclusions in Section 6.

2 Related work

In this section we review related work on large-scale image classification, metric learning, and transfer learning.

2.1 Large-scale image classification

The ImageNet dataset [6] has been a catalyst for research on large-scale image annotation. The current state-of-the-art [1], [2] uses efficient linear SVM classifiers trained in a one-vs-rest manner in combination with high-dimensional bag-of-words [18], [19] or Fisher vector representations [16]. Besides one-vs-rest training, large-scale ranking-based formulations have also been explored in [3]. Interestingly, their WSABIE approach performs joint classifier learning and dimensionality reduction of the image features. Operating in a lower-dimensional space acts as a regularization during learning, and also reduces the cost of classifier evaluation at test time. Our proposed NCM approach also learns low-dimensional projection matrices but the weight vectors are constrained to be the projected class means. This allows for efficient addition of novel classes.

In [3], [7] k-NN classifiers were found to be competitive with linear SVM classifiers in a very large-scale setting involving 10,000 or more classes. The drawback of k-NN classifiers, however, is that they are expensive in storage and computation, since in principle all training data needs to be kept in memory and accessed to classify new images. This holds even more for Naive-Bayes Nearest Neighbor (NBNN) [9], which does not use descriptor quantization, but requires storage of all local descriptors of all training images. The storage issue is also encountered when SVM classifiers are trained since all training data needs to be processed in multiple passes. Product quantization (PQ) was introduced in [15] as a lossy compression mechanism for local SIFT descriptors in a bag-of-features image retrieval system. It has been subsequently used to compress bag-of-words and Fisher vector image representations in the context of image retrieval [20] and classifier training [1]. We also exploit PQ encoding in our work to compress high-dimensional image signatures when learning our metrics.

2.2 Metric learning

There is a large body of literature on metric learning, but here we limit ourselves to highlighting just several methods that learn metrics for (image) classification problems. Other methods aim at learning metrics for verification problems and essentially learn binary classifiers that threshold the learned distance to decide whether two images belong to the same class or not, see *e.g.* [21], [22], [23]. Yet another line of work concerns metric learning for ranking problems, *e.g.* to address text retrieval tasks as in [24].

Among those methods that learn metrics for classification, the Large Margin Nearest Neighbor (LMNN) approach of [11] is specifically designed to support k-NN classification. It tries to ensure that for each image a predefined set of target neighbors from the same class are closer than samples from other classes. Since the cost function is defined over triplets of points —that can be sampled in an SGD training procedure— this method can scale to large datasets. The set of target neighbors is chosen and fixed using the ℓ_2 metric in the original space; this can be problematic as the ℓ_2 distance might be quite different from the optimal metric for image classification. Therefore, we explore two variants of LMNN that avoid using such a pre-defined set of target neighbors, similar to the ideas presented in [12].

The large margin nearest local mean classifier [25] assigns a test image to a class based on the distance to the mean of its nearest neighbors in each class. This method was reported to outperform LMNN but requires computing all pairwise distances between training instances and therefore does not scale well to large datasets. Similarly, TagProp [8] suffers from the same problem; it consists in assigning weights to training samples based on their distance to the test instance and in computing the class prediction by the total weight of samples of each class in a neighborhood.

Other closely related methods are metric learning by collapsing classes [26] and neighborhood component analysis [27]. As TagProp, for each data point these define weights to other data points proportional to the exponent of negative distance. In [26] the target is to learn a distance that makes the weights uniform for samples of the same class and close to zero for other samples. While in [27] the target is only to ensure that zero weight is assigned to samples from other classes. These methods also require computing distances between all pairs of data points. Because of their poor scaling, we do not consider any of these methods below.

Closely related to our NCMML metric learning approach for the NCM classifier is the LESS model of [28]. They learn a diagonal scaling matrix to modify the ℓ_2 distance by rescaling the data dimensions, and include an ℓ_1 penalty on the weights to perform feature selection. However, in their case, NCM is used to address small sample size problems in binary classification, *i.e*. cases where there are fewer training points (tens to hundreds) than features (thousands). Our approach differs significantly in that (i) we work in a multi-class setting and (ii) we learn a low-dimensional projection which allows efficiency in large-scale.

Another closely related method is the Taxonomy-embedding method of [29], where a nearest prototype classifier is used in combination with a hierarchical cost function. Documents are embedded in a lower dimensional space in which each class is represented by a single prototype. In contrast to our approach, they use a predefined embedding of the images and learn low-dimensional classifies, and therefore their method resembles more to the WSABIE method of [3].

The Sift-bag kernel of [30] is also related to our method since it uses an NCM classifier and an ℓ_2 distance in a subspace that is orthogonal to the subspace with maximum within-class variance. However, it involves computing the first eigenvectors of the within-class covariance matrix, which has a computational cost between $O(D^2)$ and $O(D^3)$, undesirable for high-dimensional feature vectors. Moreover, this metric is heuristically obtained, rather than directly optimized for maximum classification performance.

Finally, the image-to-class metric learning method of [31], learns per class a Mahalanobis metric, which in contrast to our method cannot generalize to new classes. Besides, it uses the idea of NBNN [9], and therefore requires the storage of all local descriptors of all images, which is impractical for the large-scale datasets used in this paper.

2.3 Transfer learning

The term transfer learning is used to refer to methods that share information across classes during learning. Examples of transfer learning in computer vision include the use of part-based or attribute class representations. Part-based object recognition models [32] define an object as a spatial constellation of parts, and share the part detectors across different classes. Attribute-based models [33] characterize a category (*e.g*. a certain animal) by a combination of attributes (*e.g*. is yellow, has stripes, is carnivore), and share the attribute classifiers across classes. Other approaches include biasing the weight vector learned for a new class towards the weight vectors of classes that have already been trained [34]. Zero-shot learning [35] is an extreme case of transfer learning where for a new class no training instances are available but a description is provided in terms of parts, attributes, or other relations to already learned classes. Transfer learning is related to multi-task learning, where the goal is to leverage the commonalities between several distinct but related classification problems, or classifiers learned for one type of images (*e.g*. ImageNet) are adapted to a new domain (*e.g*. imagery obtained from a robot camera), see *e.g*. [36], [37].

In [38] various transfer learning methods were evaluated in a large-scale setting using the ILSVRC'10 dataset. They found transfer learning methods to have little added value when training images are available for all classes. In contrast, transfer learning was found to be effective in a zero-shot learning setting, where classifiers were trained for 800 classes, and performance was tested in a 200-way classification across the held-out classes.

In this paper we also aim at transfer learning, in the sense that we allow only a trivial amount of processing on the data of new classes (storing in a database, or averaging), and rely on a metric that was trained on other classes to recognize the new ones. In contrast to most works on transfer learning, we do not use any intermediate representation in terms of parts or attributes, nor do we train classifiers for the new classes. While also considering zero-shot learning, we further evaluate performance when combining a zero-shot model inspired by [38] with progressively more training images per class, from one up to thousands. We find that the zero-shot model provides an effective prior when a small amount of training data is available.

3 The Nearest Class Mean Classifier

The nearest class mean (NCM) classifier assigns an image to the class $c^* \in \{1, \dots, C\}$ with the closest mean:

$$c^* = \operatorname*{argmin}_{c \in \{1,\dots,C\}} d(\boldsymbol{x}, \boldsymbol{\mu}_c), \tag{1}$$

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i:y_i=c} \boldsymbol{x}_i, \tag{2}$$

where $d(\boldsymbol{x}, \boldsymbol{\mu}_c)$ is the Euclidean distance between an image $\boldsymbol{x}$ and the class mean $\boldsymbol{\mu}_c$, and y_i is the ground-truth label of image i, and N_c is the number of training images in class c.

Next, we introduce our NCM metric learning approach, and its relations to existing models. Then, we present an extension to use multiple centroids per class, which transforms the NCM into a non-linear classifier. Finally, we explore some variants of the objective which allow for smaller SGD batch sizes, and we give some insights in the critical points of the objective function.

3.1 Metric learning for the NCM classifier

In this section we introduce our metric learning approach, which we will refer to as "nearest class mean metric learning" (NCMML). We replace the Euclidean distance in NCM by a learned (squared) Mahalanobis distance:

$$d_M(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^\top M (\boldsymbol{x} - \boldsymbol{x}'), \tag{3}$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are D dimensional vectors, and M is a positive definite matrix. We focus on low-rank metrics with $M = W^\top W$ and $W \in \mathbb{R}^{d \times D}$, where $d \leq D$ acts as regularizer and improves efficiency for computation and storage. The Mahalanobis distance induced by W is equivalent to the squared ℓ_2 distance after linear projection of the feature vectors on the rows of W:

$$\begin{aligned} d_W(\boldsymbol{x}, \boldsymbol{x}') &= (\boldsymbol{x} - \boldsymbol{x}')^\top W^\top W (\boldsymbol{x} - \boldsymbol{x}') \\ &= \| W\boldsymbol{x} - W\boldsymbol{x}' \|_2^2 \,. \end{aligned} \tag{4}$$

We do not consider using the more general formulation of $M = W^\top W + S$, where S is a diagonal matrix, as in [24]. While this formulation requires only D additional parameters to estimate, it still requires computing distances in the original high-dimensional space. This is costly for the dense and high-dimensional (4K-64K) Fisher vectors representations we use in our experiments, see Section 5.

We formulate the NCM classifier using a probabilistic model based on multi-class logistic regression and define the probability for a class c given an feature vector $\boldsymbol{x}$ as:

$$p(c|\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2} d_W(\boldsymbol{x}, \boldsymbol{\mu}_c)\right)}{\sum_{c'=1}^{C} \exp\left(-\frac{1}{2} d_W(\boldsymbol{x}, \boldsymbol{\mu}_{c'})\right)}. \tag{5}$$

This definition may also be interpreted as giving the posterior probabilities of a generative model where $p(\boldsymbol{x}_i|c) = \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_c, \Sigma)$, is a Gaussian with mean $\boldsymbol{\mu}_c$, and a covariance matrix $\Sigma = \left(W^\top W\right)^{-1}$, which is shared across all classes[1]. The class probabilities $p(c)$ are set to be uniform over all classes. Later, in Eq. (21), we formulate an NCM classifier with non-uniform class probabilities.

To learn the projection matrix W, we maximize the log-likelihood of the correct predictions of the training images:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \ln p(y_i|\boldsymbol{x}_i). \tag{6}$$

The gradient of the NCMML objective Eq. (6) is:

$$\nabla_W \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \alpha_{ic} \, W \, \boldsymbol{z}_{ic} \boldsymbol{z}_{ic}^\top, \tag{7}$$

1. Strictly speaking the covariance matrix is not properly defined as the low-rank matrix $W^\top W$ is non-invertible.

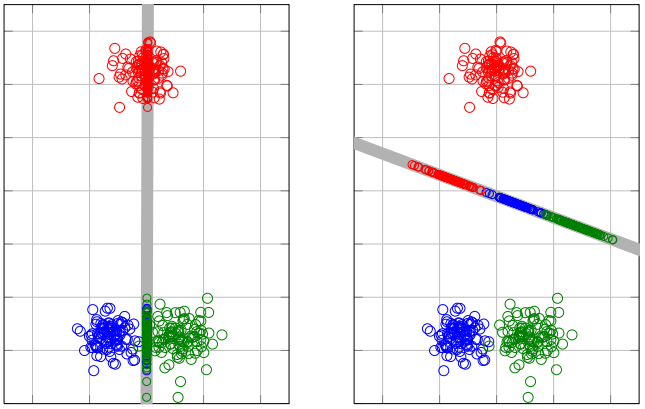

Fig. 1: Illustration to compare FDA (*left*) and NCMML (*right*), the obtained projection direction is indicated by the gray line on which also the projected samples are plotted. For FDA the result is clearly suboptimal since the blue and green classes are collapsed in the projected space. The proposed NCMML method finds a projection direction which separates the classes reasonably well.

where $\alpha_{ic} = p(c|\boldsymbol{x}_i) - [\![y_i = c]\!]$, $\boldsymbol{z}_{ic} = \boldsymbol{\mu}_c - \boldsymbol{x}_i$, and we use the Iverson brackets $[\![\cdot]\!]$ to denote the indicator function that equals one if its argument is true and zero otherwise.

Although not included above for clarity, the terms in the log-likelihood in Eq. (6) could be weighted in cases where the class distributions in the training data are not representative for those when the learned model is applied.

3.2 Relation to existing linear classifiers

First we compare the NCMML objective with the classic Fisher Discriminant Analysis (FDA) [10]. The objective of FDA is to find a projection matrix W that maximizes the ratio of between-class variance to within-class variance:

$$L_{\text{FDA}} = \operatorname{tr}\left(\frac{W S_{\text{B}} W^\top}{W S_{\text{W}} W^\top}\right), \tag{8}$$

where $S_{\text{B}} = \sum_{c=1}^{C} \frac{N_c}{N} (\boldsymbol{\mu} - \boldsymbol{\mu}_c)(\boldsymbol{\mu} - \boldsymbol{\mu}_c)^\top$ is the weighted covariance matrix of the class centers ($\boldsymbol{\mu}$ being the data center), and $S_{\text{W}} = \sum_{c=1}^{C} \frac{N_c}{N} \Sigma_c$ is the weighted sum of within class covariance matrices Σ_c, see *e.g.* [10] for details.

In the case where the within class covariance for each class equals the identity matrix, the FDA objective seeks the direction of maximum variance in S_{B}, *i.e.* it performs a PCA projection on the class means. To illustrate this, we show an example of a two-dimensional problem with three classes in Figure 1. In contrast, our NCMML method aims at separating the classes which are nearby in the projected space, so as to ensure correct predictions. The resulting projection separates the three classes reasonably well.

To relate the NCM classifier to other linear classifiers, we represent them using the class specific score functions:

$$f(c, \boldsymbol{x}) = \boldsymbol{w}_c^\top \boldsymbol{x} + b_c, \tag{9}$$

which are used to assign samples to the class with maximum score. NCM can be recognized as a linear classifier by

defining f_{NCM} with bias and weight vectors given by:

$$b_c = -\tfrac{1}{2} \| W\boldsymbol{\mu}_c \|_2^2, \tag{10}$$

$$\boldsymbol{w}_c = W^\top W\boldsymbol{\mu}_c. \tag{11}$$

This is because $-\frac{1}{2}d_W(\boldsymbol{x}, \boldsymbol{\mu}_c)$ in Eq. (5) can be written as:

$$-\ \tfrac{1}{2}\| W\boldsymbol{x} \|_2^2 - \tfrac{1}{2}\| W\boldsymbol{\mu}_c \|_2^2 + \boldsymbol{x}^\top W^\top W\boldsymbol{\mu}_c,$$

where the first term is independent of the class c and therefore irrelevant for classification.

These definitions allows us to relate the NCM classifier to other linear methods. For example, we obtain standard multi-class logistic regression, if the restrictions on b_c and $\boldsymbol{w}_c$ are removed. Note that these are precisely the restrictions that allow us adding new classes at near-zero cost, since the class specific parameters b_c and $\boldsymbol{w}_c$ are defined by just the class means $\boldsymbol{\mu}_c$ and the class-independent projection W.

In WSABIE [3] f_{WSABIE} is defined using $b_c = 0$ and,

$$\boldsymbol{w}_c = W^\top \boldsymbol{v}_c, \tag{12}$$

where $W \in \mathbb{R}^{d\times D}$ is also a low-rank projection matrix shared between all classes, and $\boldsymbol{v}_c$ is a class specific weight vector of dimensionality d, both learned from data. This is similar to NCM if we set $\boldsymbol{v}_c = W\boldsymbol{\mu}_c$. As in multiclass logistic regression, however, for WSABIE the $\boldsymbol{v}_c$ need to be learned from scratch for new classes.

The NCM classifier can also be related to the solution of ridge-regression (RR, or regularized linear least-squares regression), where the parameters b_c and $\boldsymbol{w}_c$ are learned by optimizing the squared loss:

$$L_{\text{RR}} = \frac{1}{N}\sum_i \Big(f_{\text{RR}}(c, \boldsymbol{x}_i) - y_{ic} \Big)^2 + \lambda \| \boldsymbol{w}_c \|_2^2, \tag{13}$$

where λ acts as regularizer, and where $y_{ic} = 1$, if image i belongs to class c, and $y_{ic} = 0$ otherwise. The loss L_{RR} can be minimized in closed form and leads to:

$$b_c = \frac{N_c}{N}, \quad \text{and} \qquad \boldsymbol{w}_c = \frac{N_c}{N}\boldsymbol{\mu}_c^\top (\Sigma + \lambda I)^{-1}, \tag{14}$$

where Σ is the (class-independent) data covariance matrix. Just like the NCM classifier, the RR classifier also allows to add new classes at low cost, since the class specific parameters can be found from the class means and counts once the data covariance matrix Σ has been estimated. Moreover, if N_c is equal for all classes, RR is similar to NCM with W set such that $W^\top W = (\Sigma + \lambda I)^{-1}$.

Finally, the Taxonomy-embedding [29] scores a class by:

$$f_{\text{TAX}}(c, \boldsymbol{x}) = \boldsymbol{v}_c^\top W^\top W\boldsymbol{x} - \tfrac{1}{2} \| \boldsymbol{v}_c \|_2^2, \tag{15}$$

where $W \in \mathbb{R}^{C\times D}$ projects the data to a C dimensional space, and is set using a closed-form solution based on ridge-regression. The class-specific weight vectors $\boldsymbol{v}_c$ are learned from the data. Therefore, this method relates to the WSABIE method; it learns the classifier in low-dimensional space (if $C < D$), but in this case the projection matrix W is given in closed-form. It also shares the disadvantage of the WSABIE method: it cannot generalize to novel classes without retraining.

3.3 Non-linear NCM with multiple class centroids

In this section we extend the NCM classifier to allow for more flexible class representations, which result in non-linear classification. The idea is to represent each class by a set of centroids, instead of only the class mean.

Assume that we have a set of k centroids $\{\boldsymbol{m}_{cj}\}_{j=1}^k$ for each class c. The posterior probability for class c can be defined as:

$$p(c|\boldsymbol{x}) = \sum_{j=1}^k p(\boldsymbol{m}_{cj}|\boldsymbol{x}), \tag{16}$$

$$p(\boldsymbol{m}_{cj}|\boldsymbol{x}) = \frac{1}{Z}\exp\big(\ -\tfrac{1}{2}d_W(\boldsymbol{x}, \boldsymbol{m}_{cj})\ \big), \tag{17}$$

where $p(\boldsymbol{m}_{cj}|\boldsymbol{x})$ denotes the posterior of a centroid $\boldsymbol{m}_{cj}$, and $Z = \sum_c \sum_j \exp\big(-\frac{1}{2}d_W(\boldsymbol{x}, \boldsymbol{m}_{cj})\big)$ is the normalizer.

The value k offers a transition between NCM ($k = 1$), and a weighted k-NN (k equals all images per class), where the weight of each neighbor is defined by the soft-min of its distance, *c.f.* Eq. (17). This is similar to TagProp [8], used for multi-label image annotation.

This model also corresponds to a generative model, where the probability for a feature vector $\boldsymbol{x}$, to be generated by class c, is given by a Gaussian mixture distribution:

$$p(\boldsymbol{x}|c) = \sum_{j=1}^k \pi_{cj}\, \mathcal{N}\left(\boldsymbol{x}_i; \boldsymbol{m}_{cj}, \Sigma\right), \tag{18}$$

with equal mixing weights $\pi_{cj} = 1/k$, and the covariance matrix Σ shared among all classes. We refer to this method as the nearest class multiple centroids (NCMC) classifier. A similar model was independently developed recently for image retrieval in [39]. Their objective, however, is to discriminate between different senses of a textual query, and they use a latent model to select the sense of a query.

To learn the projection matrix W, we again maximize the log-likelihood of correct classification, for which the gradient w.r.t. W in this case is given by:

$$\nabla_W \mathcal{L} = \frac{1}{N}\sum_{i,c,j} \alpha_{icj}\ W\ \boldsymbol{z}_{icj}\boldsymbol{z}_{icj}^\top, \tag{19}$$

where $\boldsymbol{z}_{icj} = \boldsymbol{m}_{cj} - \boldsymbol{x}_i$, and

$$\alpha_{icj} = p(\boldsymbol{m}_{cj}|\boldsymbol{x}_i) - [\![c = y_i]\!] \frac{p(\boldsymbol{m}_{cj}|\boldsymbol{x}_i)}{\sum_{j'} p(\boldsymbol{m}_{cj'}|\boldsymbol{x}_i)}. \tag{20}$$

To obtain the centroids of each class, we apply k-means clustering on the features $\boldsymbol{x}$ belonging to that class, using the ℓ_2 distance. Instead of using a fixed set of class means, it could be advantageous to iterate the k-means clustering and the learning of the projection matrix W. Such a strategy allows the set of class centroids to represent more precisely the distribution of the images in the projected space, and might further improve the classification performance. However the experimental validation of such a strategy falls beyond the scope of this paper.

TABLE 1: Comparison of complexity of the considered alternatives to compute the class probabilities $p(c|\boldsymbol{x})$.

Distances in D dimensions	$O\big(dD(mC) + mC(d+D)\big)$
Distances in d dimensions	$O\big(dD(m+C) + mC(d)\big)$
Dot product formulation	$O\big(dD(m) + mC(D)\big)$

3.4 Alternative objective for small SGD batches

Computing the gradients for NCMML in Eq. (7) and NCMC in Eq. (19) is relatively expensive, regardless of the number of m samples used per SGD iteration. The cost of this computation is dominated by the computation of the squared distances $d_W(\boldsymbol{x}, \boldsymbol{\mu}_c)$, required to compute the $m \times C$ probabilities $p(c|\boldsymbol{x})$ for C classes in the SGD update. To compute these distances we have two options. First, we can compute the $m \times C$ difference vectors $(\boldsymbol{x} - \boldsymbol{\mu}_c)$, project these on the $d \times D$ matrix W, and compute the norms of the projected difference vectors, at a total cost of $O\big(dD(mC) + mC(d+D)\big)$. Second, we can first project both the m data vectors and C class centers, and then compute distances in the low dimensional space, at a total cost of $O\big(dD(m+C) + mC(d)\big)$. Note that the latter option has a lower complexity, but still requires projecting all class centers at a cost $O(dDC)$, which will be the dominating cost when using small SGD batches with $m \ll C$. Therefore, in practice we are limited to using SGD batch sizes with $m \approx C = 1,000$ samples.

In order to accommodate for fast SGD updates based on smaller batch sizes, we replace the Euclidean distance in Eq. (5) by the negative dot-product plus a class specific bias s_c. The probability for class c is now given by:

$$p(c|\boldsymbol{x}_i) = \frac{1}{Z} \exp\Big(\boldsymbol{x}_i^\top W^\top W \boldsymbol{\mu}_c + s_c \Big), \qquad (21)$$

where Z denotes the normalizer. The objective is still to maximize the log-likelihood of Eq. (6). The efficiency gain stems from the fact that we can avoid projecting the class centers on W, by twice projecting the data vectors: $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i^\top W^\top W$, and then computing dot-products in high dimensional space $\langle \hat{\boldsymbol{x}}_i, \boldsymbol{\mu}_c \rangle$. For a batch of m images, the first step costs $O(mDd)$, and the latter $O(mCD)$, resulting in a complexity of $O\big(dD(m) + mC(D)\big)$. This complexity scales linearly with m, and is lower for small batches with $m \leq d$, since in that case it is more costly to project the class vectors on W than on the double-projected data vectors $\hat{\boldsymbol{x}}_i$. For clarity, we summarize the complexity of the different alternatives we considered in Table 1.

A potential disadvantage of this approach is that we need to determine the class-specific bias s_c when data of a new class becomes available, which would require more training than just computing the data mean for the new class. However, we expect a strong correlation between the learned bias s_c and the bias based on the norm of the projected mean b_c, as shown in Figure 2.

Similarly, as for Eq. (5), we could interpret the class probabilities in Eq. (21) as being generated by a generative model where the class-conditional models $p(\boldsymbol{x}|c)$ are Gaussian with a shared covariance matrix. In this interpretation, the class specific biases s_c define class prior probabilities given by $p(c) \propto \exp\big(\frac{1}{2} \| W\boldsymbol{\mu}_c \|_2^2 + s_c\big)$. Therefore, a uniform prior is obtained by setting $s_c = -\frac{1}{2} \| W\boldsymbol{\mu}_c \|_2^2 = b_c$. A uniform prior is reasonable for the ILSVRC'10 data, since the classes are near uniform in the training and test data.

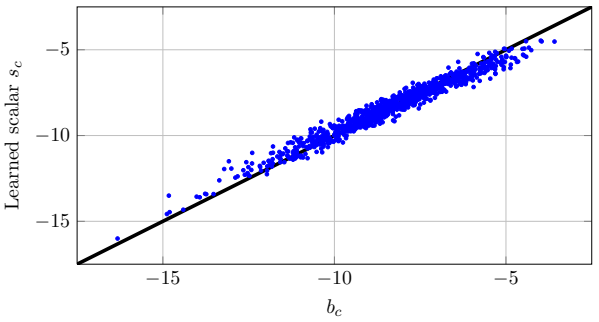

Fig. 2: The learned class-specific biases s_c and the norm of the projected means b_c are strongly correlated.

Experimentally, we find that using this formulation yields comparable results as obtained with the Euclidean distance of Eq. (5). For example, on ILSVRC'10 with 4K dimensional features and 128 dimensional projection matrix W, the classification error decreases from 39.2% when using s_c to 39.0% when using b_c at evaluation time, *c.f.* Table 4. Thus, we can use the metric learned using Eq. (21), in combination with the norm of the projected mean as bias, which is easily computed for new classes.

3.5 Critical points of low rank metric learning

We use a low-rank Mahalanobis distance where $M = W^\top W$, as a way to reduce the number of parameters and to gain in computational efficiency. Learning a full Mahalanobis distance matrix M, however, has the advantage that the distance is linear in M and that the multi-class logistic regression objective of Eq. (6) is therefore concave in M, see details in [40, page 74]. Using a low-rank formulation, on the other hand, yields a distance which is quadratic in the parameters W, therefore the objective function is no longer concave. In this section we investigate the critical-points of the low-rank formulation by analyzing W when the optimization reaches a (local) minimum, and considering the gradient for the corresponding full matrix $M = W^\top W$.

The gradient of the objective of Eq. (6) w.r.t. to M is:

$$\nabla_M \mathcal{L} = \frac{1}{N} \sum_{i,c} \alpha_{ic}\, \boldsymbol{z}_{ic} \boldsymbol{z}_{ic}^\top \equiv H, \qquad (22)$$

where $\alpha_{ic} = [\![y_i = c]\!] - p(c|\boldsymbol{x}_i)$, and $\boldsymbol{z}_{ic} = \boldsymbol{\mu}_c - \boldsymbol{x}_i$. Then Eq. (7) follows from the matrix chain rule, and we re-define $\nabla_W \mathcal{L} \equiv 2WH$. From the gradient w.r.t. W we immediately observe that $W = 0$ leads to a degenerate case to obtain a zero gradient, and similarly for each row of W. Below, we concentrate on the non-degenerate case.

We observe that H is a symmetric matrix, containing the difference of two positive definite matrices. In the analysis we use the eigenvalue decomposition of $H = V \Lambda V^\top$, with the columns of V being the eigenvectors, and the eigenvalues are on the diagonal of Λ.

We can now express the gradient for W as

$$\nabla_W \mathcal{L} = 2WV\Lambda V^\top \equiv G. \tag{23}$$

Thus the gradient of the i-th row of W, which we denote by $\boldsymbol{g}_i$, is a linear combination of the eigenvectors of H:

$$\boldsymbol{g}_i \equiv \sum_j \lambda_j \langle \boldsymbol{w}_i, \boldsymbol{v}_j \rangle \boldsymbol{v}_j, \tag{24}$$

where $\boldsymbol{w}_i$ and $\boldsymbol{v}_j$ denote the i-th row of W and the j-th column of V respectively. Thus an SGD gradient update will drive a row of W towards the eigenvectors of H that (i) have a large positive eigenvalue, and (ii) are most aligned with that row of W. This is intuitive, since we would expect the low-rank formulation to focus on the most significant directions of the full-rank metric.

Moreover, the expression for the gradient in Eq. (24) shows that at a critical point W^* of the objective function, all linear combination coefficients are zero: $\forall_{i,j} : \lambda_j \langle \boldsymbol{w}_i^*, \boldsymbol{v}_j \rangle = 0$. This indicates that at the critical point, for each row $\boldsymbol{w}_i^*$ and each eigenvector $\boldsymbol{v}_j$ it holds that either $\boldsymbol{w}_i^*$ is orthogonal to $\boldsymbol{v}_j$, or that $\boldsymbol{v}_j$ has a zero associated eigenvalue, *i.e.* $\lambda_j = 0$. Thus, at a critical point W^*, the corresponding gradient for the full rank formulation at that point, with $M^* = W^{*\top} W^*$, is zero in the subspace spanned by W^*.

Given this analysis, we believe it is unlikely to attain poor local minima using the low rank formulation. Indeed, the gradient updates for W are aligned with the most important directions of the corresponding full-rank gradient, and at convergence the full-rank gradient is zero in the subspace spanned by W. To confirm this, we have also experimentally investigated this by training several times with different random initializations of W. We observe that the classification performance differs at most $\pm 0.1\%$ on any of the error measures used in Section 5, and that the number of SGD iterations selected by the early stopping procedure are of the same order.

4 K-NN Metric Learning

We compare the NCM classifier to the k-NN classifier, a frequently used distance based classifier. For successful k-NN classification, the majority of the nearest neighbors should be of the same class. This is reflected in the LMNN metric learning objective [11], which is defined over triplets consisting of a query image q, an image p from the same class, and an image n from another class:

$$L_{qpn} = \left[1 + d_W(\boldsymbol{x}_q, \boldsymbol{x}_p) - d_W(\boldsymbol{x}_q, \boldsymbol{x}_n)\right]_+, \tag{25}$$

where $[z]_+ = \max(0, z)$. The hinge-loss for a triplet is zero if the negative image n is at least one distance unit farther from the query q than the positive image p, and the loss is positive otherwise. The final learning objective sums the losses over all triplets:

$$L_{\text{LMNN}} = \sum_q \sum_{p \in P_q} \sum_{n \in N_q} L_{qpn}, \tag{26}$$

where P_q and N_q denote a predefined set of positive and negative images for each query image q. Also in this case we could weight the terms in the loss function to account for non-representative class proportions in the training data.

Choice of target neighbors.

In the basic version of LMNN the set of targets P_q for a query q is set to the query's k nearest neighbors from the same class, using the ℓ_2 distance. The rationale is that if we ensure that these targets are closer than the instances of the other classes, then the k-NN classification will succeed. However, this implicitly assumes that the ℓ_2-targets will also be the closest points from the same class using the learned metric, which in practice might not be the case. Therefore, we consider two alternatives to using a fixed set of target neighbors.

First, we consider P_q to contain all images of the same class as q, hence the selection is independent on the metric. This is similar to [12] where the same type of loss was used to learn image similarity defined as the scalar product between feature vectors after a learned linear projection.

Second, we consider dynamically updating P_q to contain the k images of the same class that are closest to q using the current metric W, hence different target neighbors can be selected depending on the metric. This method corresponds to minimizing the loss function also with respect to the choice of P_q. A similar approach was proposed in [11], where every T iterations P_q is redefined using target neighbors according to the current metric.

Triplet sampling strategy.

Here, we describe a sampling strategy which obtains the maximal number of triplets from m images selected per SGD iteration. Using a small m is advantageous since the cost of the gradient evaluation is in large part determined by computing the projections $W\boldsymbol{x}$ of the images, and, if used, the cost of decompressing the PQ encoded signatures.

To generate triplets, we first select uniformly at random a class c that will provide the query and positive images. When P_q is set to contain all images of the same class, we sample $\frac{2}{3}m$ images from the class c, and $\frac{1}{3}m$ images across other classes. In this manner we can construct about $\frac{4}{27}m^3$ triplets, *i.e.* about 4 million triplets for $m = 300$ used in our experiments, see our technical report [41] for more details.

For other choices of P_q we do the following:

- For a fixed set of target neighbors, we still sample $\frac{1}{3}m$ negative images, and take as many query images together with their target neighbors until we obtain $\frac{2}{3}m$ images allocated for the positive class.
- For a dynamic set of target neighbors we simply select the closest neighbors among the $\frac{2}{3}m$ sampled positive images using the current metric W. Although approximate, this avoids computing the dynamic target neighbors among all the images in the positive class.

Efficient gradient evaluation.

For either choice of the target set P_q, the gradient can be computed without explicitly iterating over all triplets, by sorting the distances w.r.t. query images. The sub-gradient of the loss of a triplet is given by:

$$\nabla_W L_{qpn} = [\![L_{qpn} > 0]\!] \; 2\, W \Big(\boldsymbol{x}_{qp} \boldsymbol{x}_{qp}^\top - \boldsymbol{x}_{qn} \boldsymbol{x}_{qn}^\top \Big), \tag{27}$$

1) Sort distances w.r.t. q in ascending order; for positive images use $d_W(\boldsymbol{x}_q, \boldsymbol{x}_p) + 1$ to account for the margin.
2) Accumulate, from start to end, the number of negative images up to each position.
3) Accumulate, from end to start, the number of positive images after each position.
4) Read-off the number of hinge-loss generating triplets of image p or n.

Algorithm 1: Compute coefficients A_{qn} and A_{qp}.

where $\boldsymbol{x}_{qp} = \boldsymbol{x}_q - \boldsymbol{x}_p$, $\boldsymbol{x}_{qn} = \boldsymbol{x}_q - \boldsymbol{x}_n$. We can write the gradient w.r.t. L_{LMNN} in matrix form as:

$$\nabla_W L_{\text{LMNN}} = 2\, W\, XAX^\top, \tag{28}$$

where X contains the m feature vectors used in an SGD iteration, and A is a coefficient matrix. This shows that once A is available, the gradient can be computed in time $O(m^2)$, even if a much larger number of triplets is used.

When P_q contains all images of the same class, A can be computed from the number of loss generating triplets:

$$A_{qn} = 2\sum_p [\![L_{qpn} > 0]\!], \quad A_{pq} = -2\sum_n [\![L_{qpn} > 0]\!].$$

Once A_{qp} and A_{qn} are known, the coefficients A_{qq}, A_{pp}, and A_{nn} are obtained from the former by summing, *e.g.* $A_{qq} = \sum_p A_{qp} - \sum_n A_{qn}$, see [41] for more details.

In Algorithm 1 we describe how to efficiently compute the coefficients. The same algorithm can be applied when using a small set of fixed, or dynamic target neighbors. In particular, the sorted list allows to dynamically determine the target neighbors at a negligible additional cost. In this case only the selected target neighbors obtain non-zero coefficients, and we only accumulate the number of target neighbors after each position in step 3 of the algorithm.

The cost of this algorithm is $O(m \log m)$ per query, and thus $O(m^2 \log m)$ when using $O(m)$ query images per iteration. This is significantly faster than explicitly looping over all $O(m^3)$ triplets.

Note that while this algorithm enables fast computation of the sub-gradient of the loss, the value of the loss itself cannot be determined using this method. However, this is not a problem when using an SGD approach, as it only requires gradient evaluations, not function evaluations.

5 Experimental Evaluation

In this section we experimentally validate our models described in the previous sections. We first describe the dataset and evaluation measures used in our experiments, followed by the presentation of the experimental results.

5.1 Experimental Setup and Baseline Approach

Dataset: In most of our experiments we use the dataset of the ImageNet Large Scale Visual Recognition 2010 challenge (ILSVRC'10)[2]. This dataset contains 1.2M training images of 1,000 object classes (with between 660 to 3047 images per class), a validation set of 50K images (50 per class), and a test set of 150K images (150 per class).

In some of the experiments, we use the ImageNet-10K dataset introduced in [7], which consists of 10,184 classes from the nodes of the ImageNet hierarchy with more than 200 images. We follow [1] and use 4.5M images as training set, 50K as validation set and the rest as test set.

Image representation: We represent each image with a Fisher vector (FV) [16] computed over densely extracted 128 dimensional SIFT descriptors [42] and 96 dimensional local color features [43], both projected with PCA to 64 dimensions. FVs are extracted and normalized separately for both channels and then combined by concatenating the two feature vectors. We do not make use of spatial pyramids. In our experiments we use FVs extracted using a vocabulary of either 16 or 256 Gaussians. For 16 Gaussians, this leads to a 4K dimensional feature vector, which requires about 20GB for the 1.2M training set (using 4-byte floating point arithmetic). This fits into the RAM of our 32GB servers.

For 256 Gaussians, the FVs are 16 times larger, *i.e.* 64K dimensional, which would require 320GB of memory. To fit the data in memory, we compress the feature vectors using product quantization [14], [15]. In a nutshell, it consists in splitting the high-dimensional vector into small sub-vectors, and vector quantizing each sub-vector independently. We compress the dataset to approximately 10GB using 8-dimensional sub-vectors and 256 centroids per sub-quantizer, which allows storing each sub-quantizer index in a single byte, combined with a sparse encoding of the zero sub-vectors, see [1]. In each iteration of SGD learning, we decompress the features of a limited number of images, and use these (lossy) reconstructions to compute the gradient.

Evaluation measures: We report the average top-1 and top-5 flat error used in the ILSVRC'10 challenge. The flat error is one if the ground-truth label does not correspond to the top-1 label with highest score (or any of the top-5 labels), and zero otherwise. The motivation for the top-5 error is to allow an algorithm to identify multiple objects in an image and not being penalized if one of the objects identified was in fact present but not included in the ground truth of the image which contains only a single object category per image.

Baseline approach: For our baseline, we follow the state-of-the-art approach of [44] and learn weighed one-vs-rest SVMs with SGD, where the number of negative images in each iteration is sampled proportional to the number of positive images for that class. The proportion parameter is cross validated on the validation set. The results of the baseline can be found in Table 4 and Table 7. We observe that the performance when using the 64K dimensional features (28.0) is significantly better than the 4K ones (38.2), despite the lossy PQ compression.

In Table 4 the performance using the 64K features is slightly better than the ILSVRC'10 challenge winners [2] (28.0 vs. 28.2 flat top-5 error), and close to the results of [1] (25.7 flat top-5 error), wherein an image representation of more than 1M dimensions was used. In Table 7 our baseline shows state-of-the-art performance on ImageNet-10K when

2. See http://www.image-net.org/challenges/LSVRC/2010/index

TABLE 2: Complexity comparison of classifier training.

Training times in CPU days

	4K				64K			
Method	128	256	512	Full	128	256	512	Full
SVM				2.7				21.3
NCM	1.9	4.5	12.1		32.9	80.3	141.2	
k-NN	4.6	5.1	10.1					

Number of images seen during training

	C	I	T	Total
SVM	1,000	65	120k	7,800M
NCM	1	1,000	500k	500M
k-NN	1	300	2M	600M

using the 64K features, obtaining 78.1 vs 81.9 flat top-1 error [44]. We believe that this is due to the use of the color features, in addition to the SIFT features used in [44].

SGD training and early stopping: To learn the projection matrix W, we use SGD training and sample at each iteration a fixed number of m training images to estimate the gradient. Following [24], we use a fixed learning rate and do not include an explicit regularization term, but rather use the projection dimension d, as well as the number of iterations as an implicit form of regularization. For all experiments we proceed as follows:

1) run for a large number of iterations ($\approx$ 750K-2M),
2) validate every 50K (k-NN) or 10K (NCM) iterations,
3) select metric with lowest top-5 error.

In case of a tie, the metric with the lowest top-1 error is chosen. Similarly, all hyper-parameters, like the value of k for k-NN, are validated in this way. Unless stated otherwise, training is performed using the ILSVRC'10 training set, and validation on the provided 50K images of the validation set.

Training and testing complexity: In Table 2 we give an overview of the training times and number of images seen during training for the different algorithms. While the training times are difficult to compare due to the use of different implementations (Matlab and C/C++) and different machines, it is interesting to see that the the number of training images used to convergence is roughly of the same order for the different algorithms. We compare the methods in terms of (i) the number of models that is learned: SVM learns $C = 1,000$ different classifiers, while the NCM/k-NN methods both learn a single projection matrix ($C = 1$), (ii) the number of images I per iteration: for SVM we use 64 negative images per positive image ($I = 65$), for NCM we use $I = 1,000$, and for k-NN we use $I = 300$, and (iii) the number of iterations T.

While it is straightforward to parallelize the learning of the SVMs (*e.g.* each machine learns a single classifier), it is more complex for the proposed methods where a shared projection matrix is learned for all classes. Nevertheless, the core components of these methods can be written as matrix products (*e.g.* projections of the means or images, the gradients of the objectives, *etc.*), for which we benefit from optimized multi-threaded implementations.

At test time, evaluation of the classifiers is expensive for the k-NN classifiers, but cheap for the NCM and SVM classifiers. For the SVMs, the cost is $O(MCD)$, where C is the number of classes, D the dimensionality of the feature vector and M the number of images in the test set. The NCM classifier, can be evaluated at the same cost by pre-computing the double projection of the means, similar to the approach discussed in Section 3.4. If the dimensionality of the projection matrix d is smaller than C, then it may be more efficient to project the test images in $O(MDd)$, and to compute the distances in the projected space in $O(MCd)$.

TABLE 3: Comparison of results for different k-NN classification methods using the 4K dimensional features. For all methods, except those indicated by 'Full', the data is projected to a 128 dimensional space.

	k-NN classifiers							
	SVM	ℓ_2	ℓ_2	LMNN		All	Dynamic	
	Full	Full	+ PCA	10	20		10	20
Top-5	**38.2**	55.7	57.3	50.6	50.4	44.2	**39.7**	40.7

5.2 k-NN metric learning results

We start with an assessment of k-NN classifiers in order to select a baseline for comparison with the NCM classifier. Given the cost of k-NN classifiers, we focus our experiments on the 4K dimensional features, and consider the impact of the different choices for the set of target images P_q (see Section 4), and the projection dimensionality.

We initialize W as a PCA projection, and determine the number of nearest neighbors to be used for classification on the validation set; typically 100 to 250 neighbors are optimal.

Target selection for k-NN metric learning.

In the first experiment we compare the three different options of Section 4 to define the set of target images P_q, while learning projections to 128 dimensions. For LMNN and dynamic targets, we experimented with various numbers of targets on the validation set and found that using 10 to 20 targets yields the best results.

The results in Table 3 show that all methods lead to metrics that are better than the ℓ_2 metric in the original space, or after a PCA projection to 128 dimensions. Furthermore, we can improve over LMNN by using all within-class images as targets, or even further by using dynamic targets. The success of the dynamic target selection can be explained by the fact that among the three alternatives, the learning objective is the most closely related to the k-NN classification rule. The best performance on the flat top-5 error of 39.7 using 10 dynamic targets is, however, slightly worse than the 38.2 error rate of the SVM baseline.

Impact of projection dimension on k-NN classification.

Next, we evaluate the influence of the projection dimensionality d on the performance, by varying d between 32 and 1024. We only show results using 10 dynamic targets, since this performed best among the evaluated k-NN methods. From the results in Table 4 we see that a projection to 256 dimensions yields the lowest error of 39.0, which still remains somewhat inferior to the SVM baseline.

TABLE 4: Comparison on ILSVRC'10 of the k-NN and NCM classifiers with related methods, using the 4K and 64K dimensional features and for various projection dimensions.

	4K dimensional features						
Projection dim.	32	64	128	256	512	1024	Full
SVM baseline							**38.2**
k-NN, dynamic 10	**47.2**	**42.2**	39.7	39.0	39.4	42.4	
NCM, NCMML	49.1	42.7	**39.0**	**37.4**	**37.0**	**37.0**	
NCM, FDA	65.2	59.4	54.6	52.0	50.8	50.5	
NCM, PCA + ℓ_2	78.7	74.6	71.7	69.9	68.8	68.2	68.0
NCM, PCA + inv. cov.	75.5	67.7	60.6	54.5	49.3	46.1	43.8
Ridge-regression, PCA	86.3	80.3	73.9	68.1	62.8	58.9	54.6
WSABIE	51.9	45.1	41.2	39.4	38.7	38.5	
	64K dimensional features						
SVM baseline							**28.0**
NCMML and ℓ_2			**31.7**	31.0	30.7		63.2
WSABIE			32.2	**30.1**	**29.2**		

5.3 Nearest class mean classifier results

We now consider the performance of NCM classifiers and the related methods described in Section 3. For all experiments we use the NCM with Euclidean distance according to Eq. (5). In Table 4 we show the results.

We first consider the results for the 4K dimensional features. As observed for the k-NN classifier, also for NCM using a learned metric outperforms using the ℓ_2 distance (68.0); which is worse than using ℓ_2 distances for the k-NN classifier (55.7, see Table 3). However, unexpectedly, with metric learning we observe that our NCM classifier (37.0) outperforms the more flexible k-NN classifier (39.0), as well as the SVM baseline (38.2) when projecting to 256 dimensions or more. Our implementation of WSABIE [3] scores slightly worse (38.5) than the baseline and our NCM classifier, and does not generalize to new classes without retraining.

We also compare our NCM classifier to several other algorithms which do allow generalization to new classes. First, we consider two other supervised metric learning approaches, NCM with FDA (which leads to 50.5) and ridge-regression (which leads to 54.6). We observe that NCMML outperforms both methods significantly. Second, we consider two unsupervised variants of the NCM classifier where we use PCA to reduce the dimensionality. In one case we use the ℓ_2 metric after PCA. In the other, inspired by ridge-regression, we use NCM with the metric W generated by the inverse of the regularized covariance matrix, such that $W^\top W = (\Sigma + \lambda I)^{-1}$, see Section 3.2. We tuned the regularization parameter λ on the validation set, as was also done for ridge-regression. From these results we can conclude that, just like for k-NN, the ℓ_2 metric with or without PCA leads to poor results (68.0) as compared to a learned metric. Also, the feature whitening implemented by the inverse covariance metric leads to results (43.8) that are better than using the ℓ_2 metric, and also substantially better than ridge-regression (54.6). The results are however significantly worse than using our learned metric, in particular when using low-rank metrics.

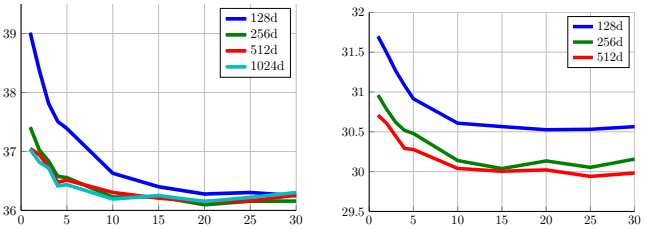

Fig. 4: Results of the NCMC-test classifier, which uses $k = 1$ at train time and $k > 1$ at test time, for the 4K (*left*) and 64K (*right*) features, for several values of k during evaluation.

TABLE 5: Results of the NCMC classifier using the 4K features, compared to the NCM classifier and the best NCMC-test classifier (with k in brackets).

	NCM	NCMC-test		NCMC		
Proj. Dim.			*(k)*	5	10	15
128	39.0	36.3	*(30)*	36.2	**35.8**	36.1
256	37.4	36.1	*(20)*	35.0	**34.8**	35.3
512	37.0	36.2	*(20)*	34.8	**34.6**	35.1

When we use the 64K dimensional features, the results of the NCM classifier (30.8) are somewhat worse than the SVM baseline (28.0); again the learned metric is significantly better than using the ℓ_2 distance (63.2). WSABIE obtains an error of 29.2, in between the SVM and NCM.

Illustration of metric learned by NCMML.

In Figure 3 we illustrate the difference between the ℓ_2 and the Mahalanobis metric induced by a learned projection from 64K to 512 dimensions. For two reference classes we show the five nearest classes, based on the distance between class means. We also show the posterior probabilities on the reference class and its five neighbor classes according to Eq. (5). The feature vector $\boldsymbol{x}$ is set as the mean of the reference class, *i.e.* a simulated perfectly typical image of this class. For the ℓ_2 metric, we used our metric learning algorithm to learn a scaling of the ℓ_2 metric to minimize Eq. (6). This does not change the ordering of classes, but ensures that we can compare probabilities computed using both metrics. We find that, as expected, the learned metric has more visually and semantically related neighbor classes. Moreover, we see that using the learned metric most of the probability mass is assigned to the reference class, whereas the ℓ_2 metric leads to rather uncertain classifications. This suggests that using the ℓ_2 metric many classes are placed at comparable distances.

Non-linear classification using multiple class centroids.

In these experiments we use the non-linear NCMC classifier, introduced in Section 3.3, where each class is represented by a set of k centroids. We obtain the k centroids per class by using the k-means algorithm in the ℓ_2 space.

Since the cost of training these classifiers is much higher, we run two sets of experiments. In Figure 4, we show the performance of the NCMC-test classifier, where only at test time $k = [2, \ldots, 30]$ is used, while using a metric obtained by the NCM objective ($k = 1$). In Table 5, we show the

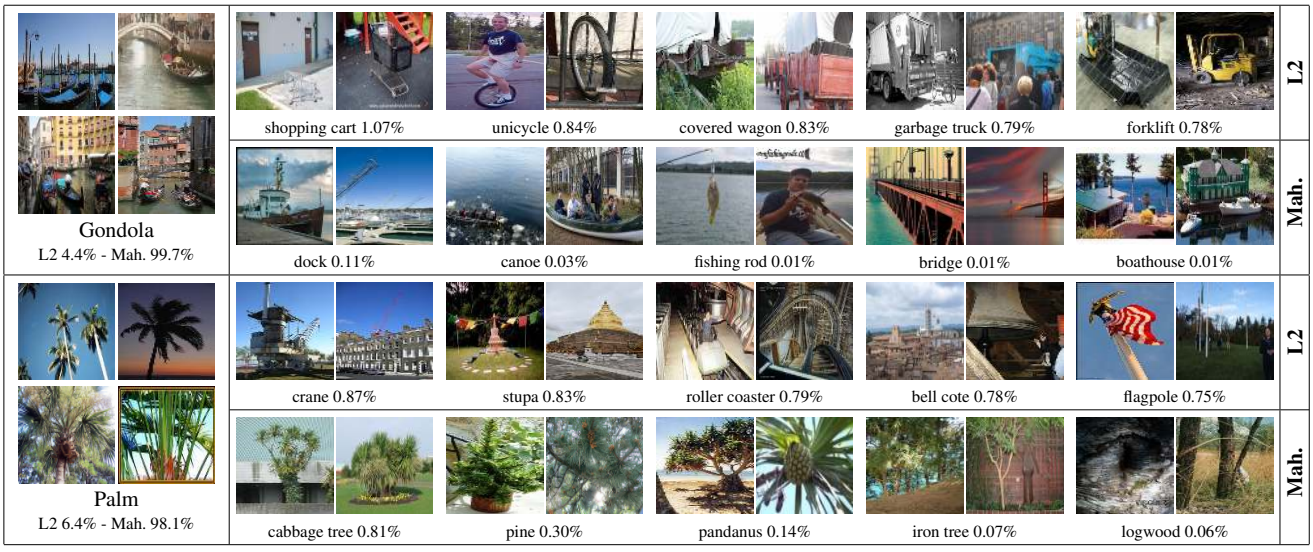

Fig. 3: The nearest classes for two reference classes using the the ℓ_2 distance and metric learned by NCMML. Class probabilities are given for a simulated image that equals the mean of the reference class, see text for details.

performance of the NCMC classifier, trained with the NCMC objective, using the 4K features, compared to the NCM method and the best NCMC-test method.

From the results we observe that a significant performance improvement can be made by using the non-linear NCMC classifier, especially when using a low number of projection dimensions. When learning using the NCMC classifier we can further improve the performance of the non-linear classification, albeit for a higher training cost. When using as little as 512 projection dimensions and $k = 10$ centroids, we obtain a performance of 34.6 on the top-5 error. This is an improvement of about 2.4 absolute points over the NCM classifier (37.0), and 3.6 absolute points over SVM classification (38.2), *c.f.* Table 4.

For the 64K features, the NCMC (with $k = 10$ and $d = 512$) yields to a top-5 error 29.4, which is about 1.3 points improvement over the NCM classifier.

5.4 Generalizing to new classes with few samples

Here we explore the ability of the distance based classifiers to generalize to novel classes. For the NCM we also consider its performance as a function of the number of training images available to estimate the mean of novel classes.

Generalization to classes not seen during training.

In this experiment we split the ILSVRC'10 dataset into a training set consisting of approximately 1M images from 800 classes, and an evaluation set of the 200 held-out classes. The error is evaluated in a 1,000-way classification task, and computed over the 30K images in the test set of the held-out classes. Performance on the test images of the 800 train classes changes only marginally and including them would obscure the changes among the test images of the 200 held-out classes. The early stopping strategy uses the validation set of the 800 training classes.

In Table 6 we show the performance of NCM and k-NN classifiers, and compare it to the control setting where the metric is trained on all 1,000 classes. The results show that both classifiers generalize remarkably well to new classes. For comparison we also include the results of the SVM baseline, and the k-NN and NCM classifiers using the ℓ_2 distance, evaluated over the 200 held-out classes. In particular for 1024 dimensional projections of the 4K features, the NCM classifier achieves an error of 40.0 over classes not seen during training, as compared to 36.5 when using all classes for training. For the 64K dimensional features the drop in performance is larger, but still surprisingly good considering that training for the novel classes consists only in computing their means.

TABLE 6: Results for 1,000-way classification among test images of 200 classes not used for metric learning, and control setting when learned on all classes.

	4K dimensional features								
	SVM	NCM					k-NN		
Projection dim.	Full	128	256	512	1024	ℓ_2	128	256	ℓ_2
Trained on all	37.6	38.6	36.8	**36.4**	36.5		39.0	38.4	
Trained on 800		42.4	40.5	40.1	**40.0**	66.6	42.7	42.6	54.2
	64K dimensional features								
Trained on all	**27.7**	31.7	30.8	30.6					
Trained on 800		39.2	38.1	**37.8**		61.9			

Generalization to the ImageNet-10K dataset: In this experiment we demonstrate the generalization ability of the NCM classifier on the ImageNet-10K dataset. We use projections learned and validated on the ILSVRC'10 dataset, and only compute the means of the 10K classes. The results in Table 7 show that, even in this extremely challenging setting, the NCM classifier performs remarkably well compared to methods which require training of 10K classifiers. We note that, to the best of our knowledge, our baseline results (78.1 top-1 error) exceed the previously known state-of-the-art (81.9 and 80.8) [44], [45].

Training our SVM baseline system took 9 and 280 CPU

TABLE 7: Comparison of the results on the ImageNet-10K dataset: the NCM classifier with metrics learned on the ILSVRC'10 dataset, the NCM using ℓ_2 distance, the baseline SVM, and previously reported SVM results [1], [7], [44] and the Deep Learning results of [45].

	4K dimensional features						64K dimensional features					Previous Results			
Method	NCM					SVM	NCM				SVM	[7]	[1]	[44]	[45]
Proj. dim.	128	256	512	1024	ℓ_2	4K	128	256	512	ℓ_2	64K	21K	131K	128K	
Top-1 error	91.8	90.6	90.5	90.4	95.5	86.0	87.1	86.3	86.1	93.6	**78.1**	93.6	83.3	81.9	80.8
Top-5 error	80.7	78.7	78.6	78.6	89.0	72.4	71.7	70.5	70.1	85.4	**60.9**				

days respectively for the 4K and 64K features, while the computation of the means for the NCM classifier took approximately 3 and 48 CPU minutes respectively. This represents roughly a 8,500 fold speed-up as compared to the SVMs, given a learned projection matrix.

Accuracy as function of sample size of novel classes.

In this experiment we consider the error as a function of the number of images that are used to compute the means of novel classes. Inspired by [38], we also include a zero-shot learning experiment, where we use the ImageNet hierarchy to estimate the mean of novel classes from related classes. We estimate the mean of a novel class $\boldsymbol{\mu}_z$ using the means of its ancestor nodes in the ILSVRC'10 class hierarchy:

$$\boldsymbol{\mu}_z = \frac{1}{|\mathcal{A}_z|} \sum_{a \in \mathcal{A}_z} \boldsymbol{\mu}_a, \tag{29}$$

where $\mathcal{A}_z$ denotes the set of ancestors of node z, and $\boldsymbol{\mu}_a$ is the mean of ancestor a. The mean of an internal node, $\boldsymbol{\mu}_a$, is computed as the average of the means of all its descendant training classes.

If we view the estimation of each class mean as the estimation of the mean of a Gaussian distribution, then the mean of a sample of images $\boldsymbol{\mu}_s$ corresponds to the Maximum Likelihood (ML) estimate, while the zero-shot estimate $\boldsymbol{\mu}_z$ can be thought of as a prior. To obtain a maximum a-posteriori (MAP) estimate $\boldsymbol{\mu}_p$, we combine the prior and the ML estimate as follows:

$$\boldsymbol{\mu}_p = \frac{n\boldsymbol{\mu}_s + m\boldsymbol{\mu}_z}{n+m}, \tag{30}$$

where the ML estimate is weighed by n the number of images that were used to compute it, and the prior mean obtains a weight m determined on the validation set [46].

In Figure 5 we analyze the performance of the NCM classifier trained on the images of the same 800 classes used above, with a learned projection from 4K and 64K to 512 dimensions. The metric and the parameter m are validated using the images of the 200 held-out classes of the validation set. We again report the error on the test images of the held-out classes in a 1,000-way classification as above. We repeat the experiment 10 times, and show error-bars at three times standard deviation. For the error to stabilize we only need approximately 100 images to estimate the class means. The results show that the zero-shot prior can be effectively combined with the empirical mean to provide a smooth transition from the zero-shot setting to a setting with many training examples. Inclusion of the zero-shot prior

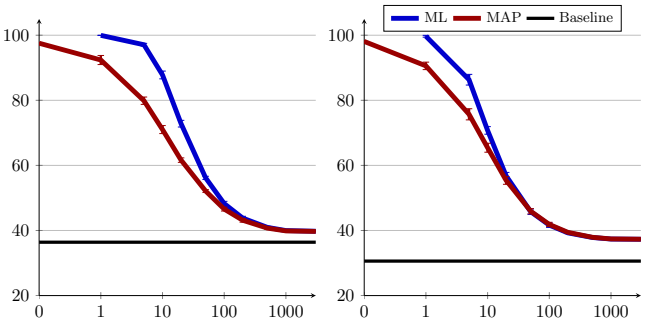

Fig. 5: Results of NCM as a function of the number of images used to compute the means for test classes. Comparison of the ML (*blue*) and MAP (*red*) mean estimates, for the 4K (*left*) and 64K (*right*) features, in a 1,000-way classification task, including baseline (*black*) when trained on all classes.

leads to a significant error reduction in the regime where ten images or less are available. Also, the results show that the validation on the 200 hold-out classes or on the 800 training classes yields comparable error rates (40.1 vs 39.9, using 4K and 512d, *c.f.* Table 6 and Figure 5).

In [38] a zero-shot error rate of 65.2 was reported in a 200-way classification task. Using the NCM with our prior mean estimates leads to comparable error rates of 66.5 (4K) and 64.0 (64K). Note that a different set of 200 hold-out classes were used, as well as different features. However their baseline performance of 37.6 top-5 error is comparable to our 4K features (38.2).

Instance level image retrieval.

Query-by-example image retrieval can be seen as an image classification problem where only a single positive sample (the query) is given and negative examples are not explicitly provided. Recently, using classifiers to learn a metric for image retrieval was considered in [47]. They found the Joint Subspace and Classifier Learning (JSCL) method to be the most effective. It consists of learning jointly a set of classifiers and a projection matrix W using WSABIE, Eq. (12) on an auxiliary supervised dataset. After training, the learned projection matrix W is used to compute distances between queries and database images.

Similarly, we propose to learn a metric using our NCM classifier on the auxiliary supervised dataset and to use the learned metric to retrieve the most similar images for a given query.

For this experiment we use the same public benchmarks as in [47]. First, the INRIA Holidays data set [48], which consists of 1,491 images of 500 scenes and objects. For

TABLE 8: Results of instance level image retrieval on the Holidays and UKB dataset, using 4K features. NCMML is compared to a PCA baseline and the JSCL method of [47].

	INRIA Holidays dataset no projection: 77.4%				UKB dataset no projection: 3.19		
Dim	PCA	JSCL	NCM	NCM*	PCA	JSCL	NCM
32	61.3	67.7	**69.3**	63.3	2.82	3.04	**3.07**
64	68.0	73.6	**75.4**	68.8	3.01	**3.23**	**3.23**
128	72.3	76.4	**79.6**	73.1	3.08	3.31	**3.33**
256	75.0	78.3	**80.2**	74.0	3.15	**3.36**	3.32
512	76.8	78.9	**80.6**	73.5	3.18	**3.36**	3.31

evaluation, one image per scene / object is used as query to search within the remaining images, and accuracy is measured as the mean average precision (mAP) over the 500 queries. Second, the University of Kentucky Benchmark dataset (UKB) [49], which contains 4 images of 2,550 objects (10,200 images). For evaluation, each image is used as query and the performance is measured by 4×recall@4 averaged over all queries, hence the maximal score is 4. For both datasets we extract the 4K image features used in our earlier experiments, these are also used in [47]. To compute the distance between two images, we use the cosine-distance, *i.e*. the dot-product on ℓ_2-normalized vectors.

We use NCMML to train a metric on the ILSVRC'10 data set, while using early stopping based on retrieval performance, similar as in [47]. To avoid tuning on the test data, the validation is performed on the other dataset, *i.e*. when testing on UKB we regularize on Holidays and *vice versa*. In Table 8 we compare the performance of the NCM based metric with that of JSCL, with a baseline PCA method, and with the performance using the original high-dimensional descriptors. Finally, for the Holidays dataset we included the NCM metric optimized for classification performance on the ILSVRC'10 validation data set (NCM*).

From these results we observe that the NCM metric yields similar performance as the JSCL method on both datasets. A projection to only 128 dimensions or more yields an equal or better retrieval performance as using the original features or the PCA baseline. On the Holidays dataset the NCM metric outperforms the JSCL metric, while on the UKB dataset JSCL slightly outperforms NCM. Both the NCM and JSCL methods are effective to learn a projection metric for instance level retrieval, while employing class level labels.

Note that it is crucial to use retrieval performance for early stopping; the results of NCM* are in fact worse than using the original descriptors. Thus, the classification objective determines a good "path" through the space of projection matrices, yet to obtain good retrieval performance the number of iterations is typically an order of magnitude smaller than for classification. We explain this discrepancy by the fact that instance level retrieval does not require the suppression of the within class variations. This suggests also that even better metrics may be learned by training NCM on a large set of queries with corresponding matches.

6 Conclusions

In this paper we have considered large-scale distance-based image classification, which allow integration of new data (possibly of new classes) at a negligible cost. This is not possible with the popular one-vs-rest SVM approach, but is essential when dealing with real-life open-ended datasets.

We have introduced NCMML, a metric learning method for NCM classification, which maximizes the log-likelihood of correct class prediction, with class probabilities using the soft-min over distances between a sample and the class means. The extended non-linear NCMC classifier offers a trade-off in the complexity, from the linear NCM to the non-parametric k-NN, by the number of used class-centroids.

We have experimentally validated our models and compared to a state-of-the-art baseline of one-vs-rest SVMs using Fisher vector image representations. Surprisingly we found that the NCM outperforms the more flexible k-NN and that its performance is comparable to a SVM baseline, while projecting the data to as few as 256 dimensions.

Our experiments on the ImageNet-10K dataset show that the learned metrics generalize well to unseen classes at a negligible cost. While only computing class means, as opposed to training 10,000 SVM classifiers, we obtain competitive performance at roughly a 8,500 fold speedup.

Finally we have also considered a zero-shot learning setting and have shown that NCM provides a unified way to treat classification and retrieval problems.

References

[1] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *CVPR*, 2011.

[2] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and SVM training," in *CVPR*, 2011.

[3] J. Weston, S. Bengio, and N. Usunier, "WSABIE: Scaling up to large vocabulary image annotation," in *IJCAI*, 2011.

[4] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *NIPS*, 2011.

[5] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *ICCV*, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009.

[7] J. Deng, A. Berg, K. Li, and L. Fei-Fei, "What does classifying more than 10,000 image categories tell us?" in *ECCV*, 2010.

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

[9] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *CVPR*, 2008.

[10] A. R. Webb, *Statistical pattern recognition*. New-York, NY, USA: Wiley, 2002.

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

[12] G. Checkik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *Journal of Machine Learning Research*, vol. 11, pp. 1109–1135, 2010.

[13] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *COMPSTAT*, 2010.

[14] R. Gray and D. Neuhoff, "Quantization," *IEEE Trans. Information Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[15] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. PAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[16] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," in *ECCV*, 2010.
[17] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Metric learning for large scale image classification: Generalizing to new classes at near-zero cost," in *ECCV*, 2012.
[18] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *ECCV Int. Workshop on Stat. Learning in Computer Vision*, 2004.
[19] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: a comprehensive study," *IJCV*, vol. 73, no. 2, pp. 213–238, 2007.
[20] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. PAMI*, 2012, to appear.
[21] E. Nowak and F. Jurie, "Learning visual similarity measures for comparing never seen objects," in *CVPR*, 2007.
[22] M. Guillaumin, J. Verbeek, and C. Schmid, "Is that you? Metric learning approaches for face identification," in *ICCV*, 2009.
[23] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," in *CVPR*, 2012.
[24] B. Bai, J. Weston, D. Grangier, R. Collobert, Y. Qi, K. Sadamasa, O. Chapelle, and K. Weinberger, "Learning to rank with (a lot of) word features," *Information Retrieval – Special Issue on Learning to Rank*, vol. 13, pp. 291–314, 2010.
[25] J. Chai, H. Liua, B. Chenb, and Z. Baoa, "Large margin nearest local mean classifier," *Signal Processing*, vol. 90, no. 1, pp. 236–248, 2010.
[26] A. Globerson and S. Roweis, "Metric learning by collapsing classes," in *NIPS*, 2006.
[27] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood component analysis," in *NIPS*, 2005.
[28] C. Veenman and D. Tax, "LESS: a model-based classifier for sparse subspaces," *IEEE Trans. PAMI*, vol. 27, no. 9, pp. 1496–1500, 2005.
[29] K. Weinberger and O. Chapelle, "Large margin taxonomy embedding for document categorization," in *NIPS*, 2009.
[30] X. Zhou, X. Zhang, Z. Yan, S.-F. Chang, M. Hasegawa-Johnson, and T. Huang, "Sift-bag kernel for video event analysis," in *ACM Multimedia*, 2008.
[31] Z. Wang, Y. Hu, and L.-T. Chia, "Image-to-class distance metric learning for image classification," in *ECCV*, 2010.
[32] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. PAMI*, vol. 28, no. 4, pp. 594–611, 2006.
[33] C. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *CVPR*, 2009.
[34] T. Tommasi and B. Caputo, "The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories," in *BMVC*, 2009.
[35] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks," in *AAAI Conference on Artificial Intelligence*, 2008.
[36] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *ECCV*, 2010.
[37] S. Parameswaran and K. Weinberger, "Large margin multi-task metric learning," in *NIPS*, 2010.
[38] M. Rohrbach, M. Stark, and B. Schiele, "Evaluating knowledge transfer and zero-shot learning in a large-scale setting," in *CVPR*, 2011.
[39] A. Lucchi and J. Weston., "Joint image and word sense discrimination for image retrieval," in *ECCV*, 2012.
[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[41] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Large scale metric learning for distance-based image classification," INRIA, Research Report RR-8077, 2012. [Online]. Available: http://hal.inria.fr/hal-00735908
[42] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[43] S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders, "XRCE's participation to ImagEval," in *ImageEval Workshop at CVIR*, 2007.
[44] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," in *CVPR*, 2012.
[45] Q. Le, M. Ranzato, R. Monga, M. Devin, K.Chen, G. Corrado, J. Dean, and A. Ng, "Building high-level features using large scale unsupervised learning," in *ICML*, 2012.
[46] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains." *IEEE Trans. Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.
[47] A. Gordo, J. Rodríguez, F. Perronnin, and E. Valveny, "Leveraging category-level labels for instance-level image retrieval," in *CVPR*, 2012.
[48] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *ECCV*, 2008.
[49] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006.

Thomas Mensink received a PhD degree in computer science in 2012 from the University of Grenoble, France, and a cum laude MSc degree in artificial intelligence from the University of Amsterdam, The Netherlands. During the research for his PhD he worked both at Xerox Research Centre Europe and at the LEAR team of INRIA Grenoble. Currently he is PostDoc at the University of Amsterdam. His research interests include machine learning and computer vision.

Jakob Verbeek received a PhD degree in computer science in 2004 from the University of Amsterdam, The Netherlands. After being a postdoctoral researcher at the University of Amsterdam and at INRIA Rhône-Alpes, he has been a full-time researcher at INRIA, Grenoble, France, since 2007. His research interests include machine learning and computer vision, with special interest in applications of statistical models in computer vision.

Florent Perronnin holds an Engineering degree from the Ecole Nationale Supérieure des Télécommunications and a PhD degree from the Ecole Polytechnique Fédérale de Lausanne. From 2000 to 2001 he was a Researcher in the Panasonic Speech Technology Laboratory working on speech and speaker recognition. In 2005, he joined the Xerox Research Centre Europe where he is currently a Principal Scientist and the Manager of the Computer Vision Group. His main interests are in the application of machine learning to problems such as image classification, retrieval and segmentation.

Gabriela Csurka obtained her Ph.D. degree in computer science in 1996 from University of Nice Sophia Antipolis. She worked in fields such as stereo vision and projective reconstruction at INRIA and image watermarking at University of Geneva and Institute Eurécom. In 2002 she joined the Xerox Research Centre Europe, where she is senior scientist. Her research interest include mono-modal and multi-modal image categorization, retrieval and semantic image segmentation.