

Large Scale Metric Learning for Distance-Based Image Classification on Open Ended Data Sets

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka

Abstract 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 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. Since the performance of distance-based classifiers heavily depends on the used distance function, we cast the problem into one of learning a low-rank metric, which is shared across all classes. For the NCM classifier we introduce a new metric learning approach, and we also introduce an extension to allow for richer class representations.

Experiments on the ImageNet 2010 challenge dataset, which contains over one million training images of thousand 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.

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

Thomas Mensink and Jakob Verbeek
LEAR Team - INRIA Grenoble, 655 Avenue de l'Europe, 38330 Montbonnot, France
e-mail: firstname.lastname@inria.fr

Florent Perronnin and Gabriela Csurka
Xerox Research Centre Europe, 6 chemin de Maupertuis, 38240 Meylan, France
e-mail: firstname.lastname@xrce.xerox.com

This chapter is largely based on previously published work [30, 31].

1 Introduction

In the last decade we have witnessed an explosion in the amount of images and videos that are digitally available, *e.g.* in broadcasting archives or social media sharing websites. Scalable automated methods are needed to handle this huge volume of data, for retrieval, annotation and visualization purposes. This need has been recognized in the computer vision research community and large-scale methods have become an active topic of research in recent years. The introduction of the ImageNet dataset [10], which contains more than 14M manually labeled images of 22K classes, has provided an important benchmark for large-scale image classification and annotation algorithms.

In this chapter 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 main object appearing in the image, or a general label like the scene type of the image. The predominant approach to this problem is to treat it as a classification problem. To ensure scalability, often linear classifiers such as linear SVMs are used [39, 27]. Additionally, to speed-up classification, dimension reduction techniques could be used [46], or a hierarchy of classifiers could be learned [2, 12]. Recently, impressive results have been reported on 10,000 or more classes [9, 46, 39]. For all these methods hold that they are trained by using stochastic gradient descent (SGD) algorithms, which access only a small fraction of the training data at each iteration [3].

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. At first glance, one-vs-rest classifiers (such as SVMs) trained with SGD algorithms appear to be the perfect solution. Indeed, classes can be trained independently, thus enabling to easily add new classes. Also since SGD algorithms are online algorithms they can accommodate for new samples easily. However, in order to apply these algorithms successfully we need to address several challenging problems. First, it is unclear to which classifier a new training sample should be fed; surely the sample should be used for the corresponding class, however feeding it as a negative sample to all other classes is not only costly but also suboptimal [35]. Second, for state-of-the-art performance the parameters of the SVM (such as the regularizer and the balance between positive and negative samples) are set by cross validation. The optimal value of these parameters depend, among others, on the number of training samples and classes, and it is unclear how these should be adapted when applied on open ended datasets to allow for an increasing number of samples and classes over time. Therefore, we believe that standard one-vs-rest SVM classifiers are unsuitable for this task.

In this work, as an alternative, we explore distance-based classifiers which enable the addition of new classes and new images to existing classes at (near) zero cost. These 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.

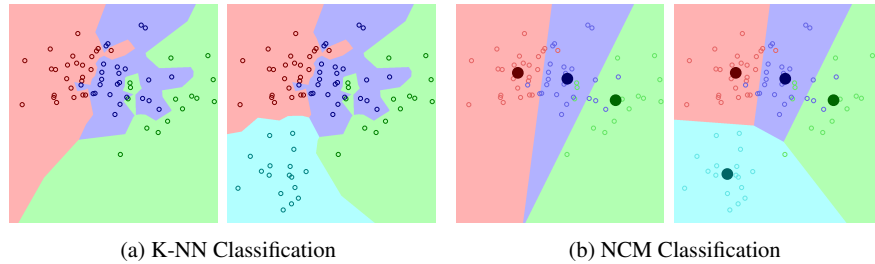


Fig. 1: Illustration of the classification rules of k-NN (a) and NCM (b). We also illustrate the effect of adding data from a new class to the data set.

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 [9, 46, 18]. New images (of new classes) are simply added to the database, and can be used for classification without further processing, see Figure 1a for an illustration.

The second is the nearest class mean classifier (NCM), which represents classes by their mean feature vector of its elements, see e.g. [42]. 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, see Figure 1b for an illustration. In Section 3, we introduce an extension which uses several prototypes per class, allowing 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 chapter 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 class mean in the projected space. We show qualitatively and quantitatively the advantages of our NCMML approach over the classical Fisher Discriminant Analysis [42]. For k-NN classification, we rely on the Large Margin Nearest Neighbor (LMNN) framework [44] and investigate two variations similar to the ideas presented in [44, 6] 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 [3]. 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 [17], a data compression technique

that was recently used with success for large-scale image retrieval [21] and classifier training [39]. As a baseline approach, we use the state-of-the-art approach of [39], which was also the winning entry in the 2011 edition of the challenge: Fisher vector image representations [36] 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, after learning a low-rank metric on the Fisher vector image representations. Moreover, the NCM classifier performs on par with the SVM baseline, and results are improved when we use the extension to allow for multiple centroids per class. Further we consider, among others, the generalization performance to new classes on the ImageNet-10K dataset (which consist of 4.5M training images of 10K classes) [9], a zero-shot setting where we estimate the mean of novel classes based on related classes in the ImageNet hierarchy, and image retrieval where we use a metric learned with our NCM classifier on the ILSVRC'10 dataset, to retrieve the most similar images for a given query on the INRIA Holidays [20] or the University of Kentucky Benchmark dataset (UKB) [32].

The rest of the chapter is organized as follows. In Section 2 we discuss a selection of related work which is most relevant to this chapter. In Section 3 we introduce the NCM classifier and the NCMML metric learning approach, together with an extension to use multiple centroids (NMC). In Section 4 we review LMNN metric learning for k-NN classifiers, and present two variants. 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 [10] has been a catalyst for research on large-scale image annotation. The current state-of-the-art [39, 27] uses efficient linear SVM classifiers trained in a one-vs-rest manner in combination with high-dimensional bag-of-words [8, 47] or Fisher vector representations [36]. Besides one-vs-rest training, large-scale ranking-based formulations have also been explored in [46]. 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 [9, 46] 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. 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 [21] 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 [22] and classifier training [39]. 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 in this section we limit ourselves to highlight 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.* [33, 19, 23]. Yet another line of work concerns metric learning for ranking problems, *i.e.* to learn a metric between a query and the documents in the database, for example to address text retrieval tasks as in [1].

Among those methods that learn metrics for classification, the Large Margin Nearest Neighbor (LMNN) approach of [43, 44] 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 [6, 44], both variants leading to significant improvements.

The large margin nearest local mean classifier [5] 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.

The TagProp method of [18] is a probabilistic nearest neighbor classifier; it consists in assigning weights to training samples based on their distance to the test in-

stance and in computing the class prediction by the total weight of samples of each class in a neighborhood. However, similar as [5], for training also TagProp requires pairwise distances between all training examples. Other closely related methods are metric learning by collapsing classes [14] and neighborhood component analysis [15]. As TagProp, for each data point these define weights to other data points proportional to the exponent of negative distance. In [14] 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 [15] 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, and therefore we do not consider any of these methods in our experiments.

Closely related to our NCMML metric learning approach for the NCM classifier is the LESS model of [41]. 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 [45], 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 classifiers, and therefore their method resembles more to the WSABIE method of [46].

The centroid-based classification method explored in [48] is also related to our method. It uses a NCM classifier and an ℓ_2 distance in a subspace that is orthogonal to the subspace with maximum within-class variance. To obtain the optimal subspace, it computes the first eigenvectors of the within-class covariance matrix, which has a computational cost between $O(D^2)$ and $O(D^3)$, this is undesirable for high-dimensional feature vectors. Moreover, this metric is heuristically obtained, rather than directly optimized for maximum classification performance.

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 [11] define an object as a spatial constellation of parts, and share the part detectors across different classes. Attribute-based models [24] 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 [40]. Zero-shot learning [25] 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. [38, 34].

In [37] 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 chapter 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 [37] 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:

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

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

where $d(\mathbf{x}, \boldsymbol{\mu}_c)$ is the Euclidean distance between an image \mathbf{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 for class c . The NCM classifier is a linear classifier, which allows for efficient evaluation at test time, see Figure 1b for an illustration.

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 learns a metric by maximizing the log-likelihood of correct classification. We will refer to our method as “nearest class mean metric learning” (NCMML). In our method we replace the Euclidean distance in NCM, Eq. (1), by a learned (squared) Mahalanobis distance:

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

where \mathbf{x} and \mathbf{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 :

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

In this chapter, we do not consider using the more general formulation of $M = W^\top W + S$, where S is a diagonal matrix, as in [1]. 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, as detailed in 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 a feature vector \mathbf{x} as:

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

This definition may also be interpreted as giving the posterior probabilities of a generative model, where $p(\mathbf{x}|c) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \Sigma)$, is a Gaussian with mean $\boldsymbol{\mu}_c$, and a covariance matrix $\Sigma = (W^\top W)^{-1}$, which is shared across all classes¹. Using Bayes rule we obtain:

$$p(c|\mathbf{x}) = \frac{p(\mathbf{x}|c)p(c)}{p(\mathbf{x})} = \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \Sigma) p(c)}{\sum_{c'} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{c'}, \Sigma) p(c')}, \quad (6)$$

$$= \frac{Z(\Sigma) \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_c)) p(c)}{\sum_{c'} Z(\Sigma) \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_{c'})) p(c')}, \quad (7)$$

$$= \frac{\exp(-\frac{1}{2}d_W(\mathbf{x}, \boldsymbol{\mu}_c))}{\sum_{c'=1}^C \exp(-\frac{1}{2}d_W(\mathbf{x}, \boldsymbol{\mu}_{c'}))}, \quad (8)$$

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

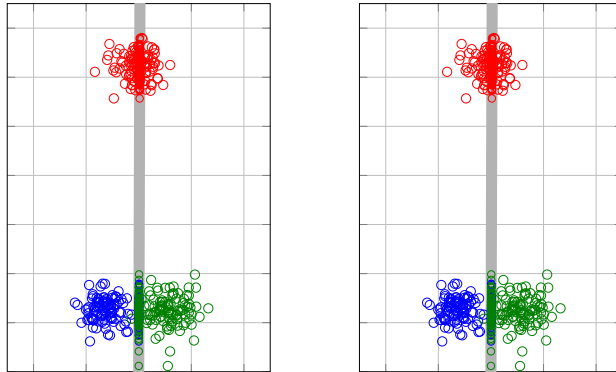


Fig. 2: Illustration to compare FDA (*left*) and NCMML (*right*), the obtained projection direction is indicated by the gray line on which also the projected samples are plotted. For FDA the result is clearly suboptimal since the blue and green classes are collapsed in the projected space. While the proposed NCMML method finds a projection which separates the classes reasonably well.

where the class probabilities $p(c)$ are set to be uniform over all classes. Later, in Eq. (27), we formulate an NCM classifier with non-uniform class probabilities.

We learn the projection matrix W in a discriminative manner, by maximizing the log-likelihood of the correct predictions of the training images:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \ln p(y_i | \mathbf{x}_i). \quad (9)$$

The gradient of the NCMML objective Eq. (9) is:

$$\nabla_W \mathcal{L} = \frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C \alpha_{ic} W \mathbf{z}_{ic} \mathbf{z}_{ic}^\top, \quad (10)$$

where $\alpha_{ic} = p(c | \mathbf{x}_i) - \mathbb{I}[y_i = c]$, $\mathbf{z}_{ic} = \boldsymbol{\mu}_c - \mathbf{x}_i$, and we use the Iverson brackets $\mathbb{I}[\cdot]$ 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. (9) 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

In this section we related the NCM classifier and the proposed NCMML approach to other linear models.

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

$$L_{\text{FDA}} = \text{tr} \left(\frac{WS_{\text{B}}W^{\top}}{WS_{\text{W}}W^{\top}} \right), \quad (11)$$

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 (and $\boldsymbol{\mu}$ is 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 . Also to obtain a well-defined problem, W has a constraint on the norms of its columns. See e.g. [42] for more details on the FDA.

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} . This equals to a PCA projection on the class means, which has the objective to maximize $\text{tr}(W^{\top}S_{\text{B}}W)$, also with a constraint on the norms of W . The result of using the unsupervised PCA technique, is that it ignores the class information in the projected space, it just maximizes the variance between all class means. To illustrate this, we show an example of a two-dimensional problem with three classes in Figure 2. In contrast to FDA, our NCMML method only aims at separating class means which are nearby in the projected space, so as to ensure correct predictions. The resulting projection direction separates the three classes reasonably well.

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

$$f(c, \mathbf{x}) = \mathbf{w}_c^{\top} \mathbf{x} + b_c, \quad (12)$$

that assigns a sample \mathbf{x} to the class with maximum score. NCM can be seen as a linear classifier by defining f_{NCM} with bias and weight vectors given by:

$$b_c = -\frac{1}{2} \|W\boldsymbol{\mu}_c\|_2^2, \quad (13)$$

$$\mathbf{w}_c = W^{\top}W\boldsymbol{\mu}_c. \quad (14)$$

This is proportional to the Mahalanobis distance up to an additive constant that is constant with respect to the class c , and therefore irrelevant for classification.

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

In the WSABIE method [46], the classifier f_{WSABIE} , is defined using $b_c = 0$ and,

$$\mathbf{w}_c = W^{\top} \mathbf{v}_c \quad (15)$$

where $W \in \mathbb{R}^{d \times D}$ is also a low-rank projection matrix shared between all classes, and \mathbf{v}_c is a class specific weight vector of dimensionality d , both learned from data.

This is similar to NCM if we set $\mathbf{v}_c = W\boldsymbol{\mu}_c$. As in multiclass logistic regression, however, for WSABIE the \mathbf{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), which also uses a linear score function. The parameters b_c and \mathbf{w}_c are learned by optimizing the squared loss:

$$L_{\text{RR}} = \frac{1}{N} \sum_i \left(f_{\text{RR}}(c, \mathbf{x}_i) - y_{ic} \right)^2 + \lambda \|\mathbf{w}_c\|_2^2, \quad (16)$$

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:

$$b_c = \frac{N_c}{N}, \quad (17)$$

$$\mathbf{w}_c = \frac{N_c}{N} \boldsymbol{\mu}_c^\top (\boldsymbol{\Sigma} + \lambda I)^{-1}, \quad (18)$$

where $\boldsymbol{\Sigma}$ is the (class-independent) data covariance matrix, see e.g. [42]. 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 $\boldsymbol{\Sigma}$ has been estimated. Moreover, if N_c is equal for all classes, RR is similar to NCM with W set such that $W^\top W = (\boldsymbol{\Sigma} + \lambda I)^{-1}$.

Finally, the Taxonomy-embedding method of [45], can be rewritten such that it equals the linear classifier f_{TAX} using:

$$b_c = -\frac{1}{2} \|\mathbf{v}_c\|_2^2, \quad (19)$$

$$\mathbf{w}_c = W^\top W \mathbf{v}_c, \quad (20)$$

where the class-specific weight vectors \mathbf{v}_c are learned from the data and $W \in \mathbb{R}^{C \times D}$ projects the data to a C dimensional space. The projection matrix W is set using a closed-form solution based on ridge-regression. This method relates to the WSABIE method since it also learns the classifier in low-dimensional space (if $C < D$), however 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 the low-dimensional class-specific vectors \mathbf{v}_c .

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, see Figure 3 for an illustration. The idea is to represent each class by a set of centroids, instead of only the class mean. Assume that we have obtained a set of k centroids $\{\mathbf{m}_{cj}\}_{j=1}^k$ for each class c . We define the posterior probability for a centroid \mathbf{m}_{cj} as:

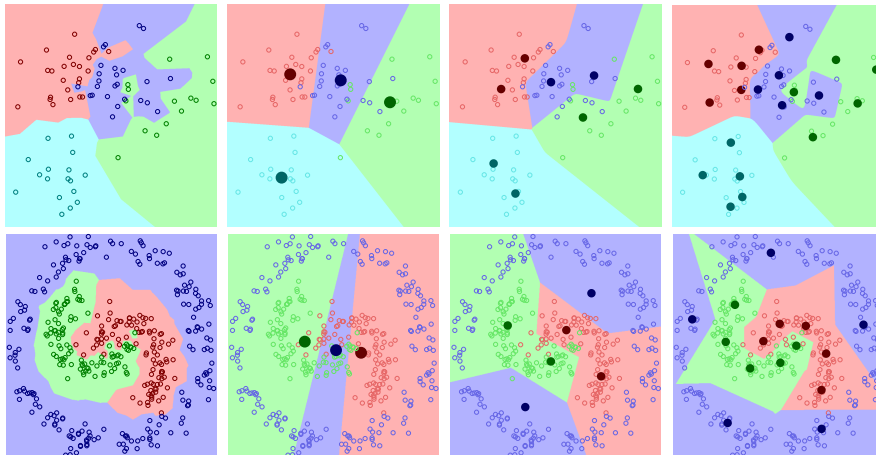


Fig. 3: Illustration of non-linear classification using multiple centroids on not linearly separable data. From left to right we show, k-NN classification ($k=1$), NCM classification, and NCMC classification with 2 and 5 centroids respectively.

$$p(\mathbf{m}_{c_j}|\mathbf{x}) = \frac{1}{Z} \exp\left(-\frac{1}{2}d_W(\mathbf{x}, \mathbf{m}_{c_j})\right), \quad (21)$$

where $Z = \sum_c \sum_j \exp\left(-\frac{1}{2}d_W(\mathbf{x}, \mathbf{m}_{c_j})\right)$ is the normalizer. The posterior probability for class c is then given by:

$$p(c|\mathbf{x}) = \sum_{j=1}^k p(\mathbf{m}_{c_j}|\mathbf{x}). \quad (22)$$

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

$$p(\mathbf{x}|c) = \sum_{j=1}^k \pi_{c_j} \mathcal{N}(\mathbf{x}; \mathbf{m}_{c_j}, \Sigma), \quad (23)$$

with equal mixing weights $\pi_{c_j} = 1/k$, 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 [29]. 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:

$$\nabla_W \mathcal{L} = \frac{1}{N} \sum_{i,c,j} \alpha_{icj} W \mathbf{z}_{icj} \mathbf{z}_{icj}^\top, \quad (24)$$

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

$$\alpha_{icj} = p(\mathbf{m}_{cj}|\mathbf{x}_i) - \llbracket c = y_i \rrbracket \frac{p(\mathbf{m}_{cj}|\mathbf{x}_i)}{\sum_{j'} p(\mathbf{m}_{cj'}|\mathbf{x}_i)}. \quad (25)$$

To obtain the centroids of each class, we apply k-means clustering on the features \mathbf{x} belonging to that class, using the ℓ_2 distance. 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. (21). This is similar to TagProp [18], used for multi-label image annotation, which assigns a probability to a class c based on the class labels and distances of the images in the training set:

$$p(c|\mathbf{x}_i) = \sum_j \pi_{ij} \llbracket y_j = c \rrbracket, \quad \pi_{ij} = \frac{\exp(-\frac{1}{2}d_W(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{j'} \exp(-\frac{1}{2}d_W(\mathbf{x}_i, \mathbf{x}_{j'}))} \quad (26)$$

In Figure 3 we illustrate the influence of increasing k on the obtained classification boundaries, and made the comparison with the k-NN ($k=1$) classifier.

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.

3.4 Alternative objective for small SGD batches

Computing the gradients for NCMML in Eq. (10) and NCMC in Eq. (24) 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 $d_W(\mathbf{x}, \boldsymbol{\mu}_c)$, required to compute the $m \times C$ probabilities $p(c|\mathbf{x})$ 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 $(\mathbf{x} - \boldsymbol{\mu}_c)$, project these on the $d \times D$ matrix W , and compute the norms of the projected difference vectors, at a total cost of $O(dD(mC) + mC(d + D))$. 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 $O(dD(m + C) + mC(d))$. 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 dot-product plus a class specific bias s_c . The probability for class c is now given by:

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

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

$$p(c|\mathbf{x}_i) = \frac{1}{Z} \exp\left(\mathbf{x}_i^\top W^\top W \boldsymbol{\mu}_c + s_c\right), \quad (27)$$

where Z denotes the normalizer. The objective is still to maximize the log-likelihood of Eq. (9). The efficiency gain stems from the fact that we can avoid projecting the class centers on W , by twice projecting the data vectors: $\hat{\mathbf{x}}_i = \mathbf{x}_i^\top W^\top W$, and then computing dot-products in high dimensional space $\langle \hat{\mathbf{x}}_i, \boldsymbol{\mu}_c \rangle$. For a batch of m images, the first step costs $O(mDd)$, and the latter $O(mCD)$, resulting in a complexity of $O(dD(m) + mC(D))$. 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 $\hat{\mathbf{x}}_i$. 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 biased based on the norm of the projected mean b_c . Similar as used for Eq. (5), we could interpret the class probabilities in Eq. (27) as those being generated by a generative model where the class-conditional models $p(\mathbf{x}|c)$ are Gaussian with a shared covariance matrix. We continue from Eq. (7) and obtain:

$$p(c|\mathbf{x}) = \frac{Z(\Sigma) \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right) p(c)}{\sum_{c'} Z(\Sigma) \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{c'})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_{c'})\right) p(c')}, \quad (28)$$

$$= \frac{\exp\left(\mathbf{x}_i^\top W^\top W \boldsymbol{\mu}_c - \frac{1}{2} \|W \boldsymbol{\mu}_c\|_2^2\right) p(c)}{\sum_{c'} \exp\left(\mathbf{x}_i^\top W^\top W \boldsymbol{\mu}_{c'} - \frac{1}{2} \|W \boldsymbol{\mu}_{c'}\|_2^2\right) p(c')} \quad (29)$$

In this interpretation, the class specific biases s_c define class prior probabilities given by $p(c) \propto \exp\left(\frac{1}{2} \|W \boldsymbol{\mu}_c\|_2^2 + s_c\right)$. 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.

Experimentally we find that using this formulation yields comparable results as using the Euclidean distance. As expected we find a strong correlation between the learned bias s_c and the norm of the projected mean b_c , shown in Figure 4. Indeed, the classification performance differs insignificantly if at evaluation time we set $s_c = b_c$ instead of the value that was found during training.

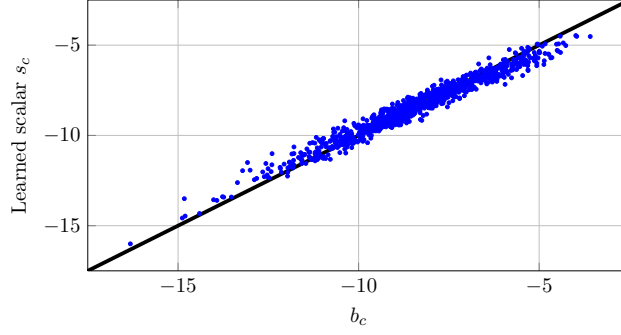


Fig. 4: The learned class-specific biases s_c and the norm of the projected means b_c are strongly correlated.

Thus, even if we train the metric by using class-specific biases, we can use the learned metric in the NCM classifier with the bias based on the norm of the projected mean, which is easily computed for data of 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. (9) is therefore concave in M [4, 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. (9) w.r.t. to M is:

$$\nabla_M \mathcal{L} = \frac{1}{N} \sum_{i,c} \alpha_{ic} \mathbf{z}_{ic} \mathbf{z}_{ic}^\top \equiv H, \quad (30)$$

where $\alpha_{ic} = p(c|\mathbf{x}_i) - \mathbb{1}[y_i = c]$, and $\mathbf{z}_{ic} = \boldsymbol{\mu}_c - \mathbf{x}_i$. Then Eq. (10) 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. Further, we observe that when $H = 0$, *i.e.* when we reach the minimum of the full Mahalanobis distance, we obtain zero gradient for W . Here we analyze H when the gradient w.r.t. W reaches a zero point. In the analysis below 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

$$\nabla_W \mathcal{L} = 2WV\Lambda V^\top \equiv G. \quad (31)$$

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

$$\mathbf{g}_i \equiv \sum_j \lambda_j \langle \mathbf{w}_i, \mathbf{v}_j \rangle \mathbf{v}_j, \quad (32)$$

where \mathbf{w}_i and \mathbf{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. (32) shows that at a critical point W^* of the objective function, all linear combination coefficients are zero: $\forall_{i,j} : \lambda_j \langle \mathbf{w}_i^*, \mathbf{v}_j \rangle = 0$. This indicates that at the critical point, for each row \mathbf{w}_i^* and each eigenvector \mathbf{v}_j it holds that either \mathbf{w}_i^* is orthogonal to \mathbf{v}_j , or that \mathbf{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.

3.6 Transfer Learning with the Nearest Class Mean Classifier

In this section we describe how we can use the NCM classifier in a zero-shot setting. Inspired by [37], we propose to use the ImageNet hierarchy to estimate the mean of novel classes from the means of related training classes, see Figure 5 for an illustration. We follow ideas of [37] and estimate the mean of a novel class $\boldsymbol{\mu}_z$ using the means of its ancestor nodes in the ImageNet class hierarchy:

$$\boldsymbol{\mu}_z = \frac{1}{|\mathcal{A}_z|} \sum_{a \in \mathcal{A}_z} \boldsymbol{\mu}_a, \quad (33)$$

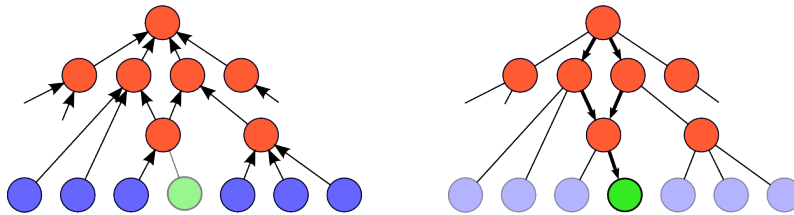


Fig. 5: Illustration of the estimation of the zero-shot prior on a mean. In the first step (*left*) the means of train classes (blue) are propagated upwards in the ImageNet hierarchy to the internal nodes (red). In the second step (*right*) the prior is estimated as the average of all the ancestor nodes of the new class (green).

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. In our experiments, the classes of interest are always leaf-nodes of the hierarchy.

In the setting where we also have a few images of the new class, we can combine the zero-shot prior with the mean of the sample images. 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. We can combine the prior with the ML estimate to obtain a maximum a-posteriori (MAP) estimate $\boldsymbol{\mu}_p$ on the class mean. The MAP estimate of the mean of a Gaussian is obtained by:

$$\boldsymbol{\mu}_p = \frac{n\boldsymbol{\mu}_s + m\boldsymbol{\mu}_z}{n + m}, \quad (34)$$

where n is the number of images used to compute the ML estimate of the sample mean $\boldsymbol{\mu}_s$, and the prior obtains a weight m determined on the validation set [13].

4 K-NN Metric Learning

We compare the NCM classifier to the k-NN classifier, a frequently used distance based classifier. The k-NN classifier is attractive since it is very intuitive, just assigning the class of the nearest neighbors to a test image, see Figure 1a for an illustration. Just as for the NCM classifier, the k-NN classifier relies on distances, and thus it is essential to use a metric in which nearby images are also semantically related. In this section we discuss metric learning for k-NN classifiers, used to learn a low-rank Mahalanobis distance $M = W^\top W$, where $W \in \mathbb{R}^{d \times D}$.

For successful k-NN classification, the majority of the nearest neighbors should be of the same class. This is reflected in the Large Margin Nearest Neighbors (LMNN) metric learning objective of [43, 44]. LMNN is defined over triplets con-

sisting of a query image q , a positive image p from the same class, and a negative image n from another class. The objective is to get the distance between q and p smaller than the distance between q and n , using the hinge-loss to upper bound the zero/one loss:

$$L_{qpn} = [1 + d_W(\mathbf{x}_q, \mathbf{x}_p) - d_W(\mathbf{x}_q, \mathbf{x}_n)]_+, \quad (35)$$

where $[z]_+ = \max(0, z)$ is the positive part of 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:

$$L_{\text{LMNN}} = \sum_{q=1}^N \sum_{p \in P_q} \sum_{n \in N_q} L_{qpn}, \quad (36)$$

where P_q and N_q denote a predefined set of positive and negative images for each query image q . An important design choice is how to set P_q and N_q for each query. For the set of negative images N_q , we follow [44] and use all images not belonging to the class of the query image. Below, we describe several variants for the set of positive images P_q .

Also in this case we can weight the terms in the loss function to account for non-representative class proportions in the training data.

4.1 Choice of target neighbors

For LMNN a set of target P_q for a query q is set to some images from the same class. 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. To select the set of targets we consider three alternatives:

1. In the basic version of LMNN the set of targets P_q is set to the query's k nearest neighbors from the same class, using the ℓ_2 distance. Since this selection method tries to ensure that the ℓ_2 -targets will also be the closest points using the learned metric, it implicitly assumes that the ℓ_2 distance in the original space is a good similarity measure. In practice, however, this might not be the case.
2. The set of targets P_q is defined to contain all images of the same class as q , hence the selection is independent on the metric. This is similar to [6] 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.
3. The set of targets P_q is dynamically updated 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 current metric. This method corresponds to minimizing the loss function also with respect to the choice of P_q . A simi-

lar approach has been proposed in [44], where every T iterations P_q is redefined using target neighbors according to the current metric.

A potential disadvantage of the the last method is that it requires frequent re-computation of the target neighbors. However, below we describe an efficient gradient evaluation method, which allows to approximate the dynamic set of P_q at each iteration at a negligible additional cost compared to using a fixed set of target neighbors or using all images of the same class as targets.

Next, we will discuss an efficient triplet sampling and gradient evaluation algorithms to increase the efficiency of the SGD training.

4.2 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 $W\mathbf{x}$ of the images, and the cost of decompressing the PQ encoded signatures, if these are used.

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 ρm images from class c , with $0 < \rho < 1$, and the remaining $(m - \rho m)$ images are uniformly sampled from the other classes. We can consider the number of triplets t we can generate as a function of ρ for a given ‘budget’ of m images to be accessed. In the case where P_q is set to contain all images from the same class, the number of triplets t we can generate for a specific ρ is given by:

$$t(\rho) = (\rho m)(\rho m - 1)(m - \rho m), \quad (37)$$

since we can pair the ρm images with the $\rho m - 1$ other images from the same class, and each pair forms a triplet with any of the $m - \rho m$ negative sampled images. The number of triplets t can be approximated by:

$$t(\rho) \approx m^3 \rho^2 (1 - \rho), \quad (38)$$

and hence, the number of triplets is maximized when we choose $\rho \approx \frac{2}{3}$, in which case we can construct about $\frac{4}{27}m^3$ triplets. In our experiments, we use $\rho = \frac{2}{3}$ and $m = 300$ images per iteration, leading to about 4 million triplets per iteration.

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.

An alternative to obtain roughly 4 million triplets is to sample two images from each of the $C = 1,000$ classes. In this case, there are two query images per class, each forming a pair with the other positive image, and each pair can form a triplet with the $2(C - 1)$ images of other classes, leading to $4C(C - 1) \approx 4$ million triplets. A potential advantage of this method is that the gradient is computed in each iteration from triplets generated using all possible combinations of classes, and therefore, the gradient might be more informative. However, this sampling strategy does not allow for fast approximation of the dynamic neighbors, and we would need to access $m = 2,000$ images, which is about 7 times more costly than using the described approach with $m = 300$.

4.3 Efficient gradient evaluation.

For either choice of the target set P_q , the gradient can be computed without explicitly iterating over all triplets. In this section we introduce an efficient gradient evaluation method, which uses sorting of the distances w.r.t. query images.

The sub-gradient of the loss of a triplet is given by:

$$\nabla_W L_{qpn} = \llbracket L_{qpn} > 0 \rrbracket 2W \left(\mathbf{x}_{qp} \mathbf{x}_{qp}^\top - \mathbf{x}_{qn} \mathbf{x}_{qn}^\top \right), \quad (39)$$

where $\mathbf{x}_{qp} = \mathbf{x}_q - \mathbf{x}_p$, and $\mathbf{x}_{qn} = \mathbf{x}_q - \mathbf{x}_n$. By observing that the gradient takes the form of outer products of the feature vectors, we can write the gradient w.r.t. $L_q = \sum_{p,n} L_{qpn}$ in matrix form as:

$$\nabla_W L_q = 2W XAX^\top, \quad (40)$$

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, the gradient per query can be rewritten as:

$$\begin{aligned} \nabla_W L_q = & +2W \sum_p \left(\sum_n \llbracket L_{qpn} > 0 \rrbracket \right) (\mathbf{x}_p \mathbf{x}_p^\top - \mathbf{x}_q \mathbf{x}_p^\top - \mathbf{x}_p \mathbf{x}_q^\top) \\ & - 2W \sum_n \left(\sum_p \llbracket L_{qpn} > 0 \rrbracket \right) (\mathbf{x}_n \mathbf{x}_n^\top - \mathbf{x}_q \mathbf{x}_n^\top - \mathbf{x}_n \mathbf{x}_q^\top). \end{aligned} \quad (41)$$

Which shows that the coefficient matrix A can be computed from the number of hinge-loss generating triplets in which each $p \in P_q$ and each $n \in N_q$ for a query q occurs:

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

1. For positive images redefine $d_W(\mathbf{x}_q, \mathbf{x}_p) \leftarrow d_W(\mathbf{x}_q, \mathbf{x}_p) + 1$ to account for the margin.
2. Sort distances w.r.t. q in ascending order.
3. $C_n(i) \leftarrow \sum_{j=1}^i \llbracket j \in N_q \rrbracket$, the number of negative images up to each position.
4. $C_p(i) \leftarrow \sum_{j=i+1}^m \llbracket j \in P_q \rrbracket$, the number of positive images after each position.
5. Read-off the number of hinge-loss generating triplets of image p or n :

$$A_{qn} = -2 C_p(\text{rnk}(q, n)) \qquad A_{qp} = 2 C_n(\text{rnk}(q, p)),$$

where $\text{rnk}(q, p)$ indicates the rank of document p for the query q , and similar for $\text{rnk}(q, n)$.

$$A_{qn} = 2 \sum_p \llbracket L_{qpn} > 0 \rrbracket, \qquad A_{pq} = -2 \sum_n \llbracket L_{qpn} > 0 \rrbracket, \qquad (42)$$

$$A_{qq} = \sum_p A_{qp} - \sum_n A_{qn}, \qquad A_{pp} = \sum_q A_{qp}, \qquad A_{nn} = \sum_q A_{qn}. \qquad (43)$$

In Algorithm 1 we describe how to efficiently compute the coefficients, which sorts the distances w.r.t. the query q and then can read off the number of hinge-loss generating triplets. 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.

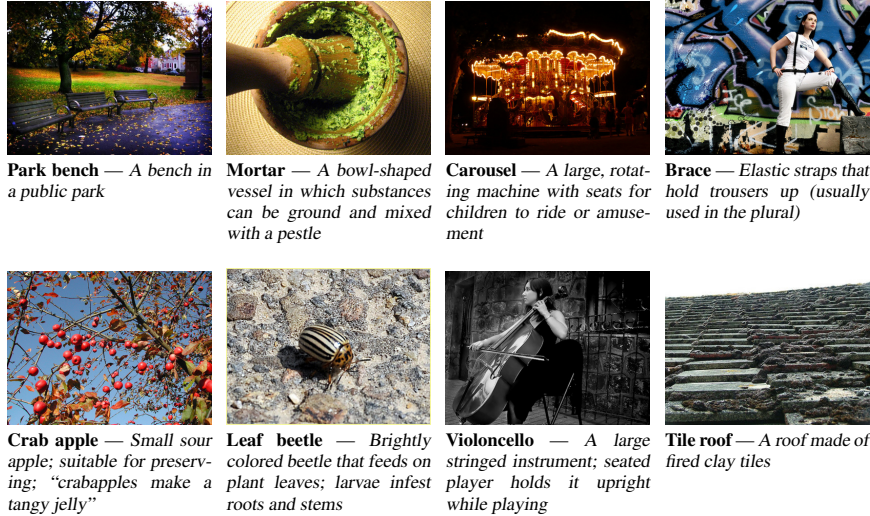


Fig. 6: Example images from the ILSVRC’10 data set, with their class names and descriptions. The data set contains 1.2M training images of 1,000 different classes.

5.1 Experimental Setup and Baseline Approach

In this section we describe the experimental setup, our image representations and our baseline methods.

Dataset. In most of our experiments we use the dataset of the ImageNet Large Scale Visual Recognition 2010 challenge (ILSVRC’10)², see Figure 6 for a few examples. 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 [9], which consists of 10,184 classes from the nodes of the ImageNet hierarchy with more than 200 images. We follow [39] 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) [36] computed over densely extracted 128 dimensional SIFT descriptors [28] and 96 dimensional local color features [7], 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 re-

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

quires 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 [17, 21]. 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, *c.f.* [39]. In each iteration of SGD learning, we decompress the features of a limited number of images, and use these (lossy) reconstructions for the gradient computation.

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. Unless specified otherwise, we report the top-5 flat error on the test set using the 4K dimensional features; we use the validation set for parameter tuning only.

Baseline approach. For our baseline, we follow the state-of-the-art approach of [35] 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 3 and Table 6. We see that the 64K dimensional features lead to significantly better results than the 4K ones, despite the lossy PQ compression.

In Table 3 the performance using the 64K features is slightly better than the ILSVRC’10 challenge winners [27] (28.0 vs. 28.2 flat top-5 error), and very close to the results of [39] (25.7 flat top-5 error), wherein a much higher dimensional image representation of more than 1M dimensions was used. In Table 6 our baseline shows state-of-the-art performance on ImageNet-10K when using the 64K features, obtaining 78.1 vs 81.9 flat top-1 error [35]. We believe this is due to the use of the color features, in addition to the SIFT features used in [35].

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 [1], 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 use the following early stopping strategy:

- we run SGD training for a large number of iterations ($\approx 750\text{K}-2\text{M}$),

- the performance on the validation set is computed every 50k iterations (for the k-NN) or every 10k iterations (for the NCM), and
- the metric which yields the lowest top-5 error on the validation set is selected.

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

It is interesting to notice that while the compared methods (k-NN, NCM, and SVM) have different computational complexities, the number of images seen by each algorithm before convergence is rather similar. For example, training of the SVMs, on the 4K features, converge after $T \approx 100$ iterations, and each iteration takes about 64 negative images per positive image, per class. In the ILSVRC'10 dataset, each class has roughly $p = 1,200$ positive images, and consist of $C = 1,000$ classes. Therefore the total number of images seen during training of the SVMs is $TC(65p) = 7.800M$ images. The NCM classifier requires much more iterations, $T \approx 500K$, but uses at each iteration only $m = 1,000$ images, and trains only a single metric. Therefore the total number of images seen during training is roughly $Tm = 500M$. Finally, the k-NN classifier, requires even more iterations, $T \approx 2M$, but uses only $m = 300$ images per iteration, the total number of images seen before convergence is therefore about $Tm = 600M$.

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 use for classification on the validation set. Typically using 100 to 250 neighbors is optimal, which is rather large for k-NN classification, for example in [44] $k = 3$ is used, and indicates that the classification function is rather smooth.

5.2.1 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 2 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.

	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

Table 2: Comparison of flat error for different k-NN classification methods using 4K dimensional features. For all methods, except those indicated by ‘Full’, the data is projected to a 128 dimensional space.

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.

5.2.2 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 3 we see that a projection to 256 dimensions yields the lowest error of 39.0, which still remains somewhat inferior to the SVM baseline.

5.3 Nearest class mean classifier results

We now consider the performance of NCM classifiers and the related methods described in Section 3. In Table 3 we show the results for various projection dimensionalities.

We first consider the results for the 4K dimensional features. As observed for the k-NN classifier, using a learned metric outperforms using the ℓ_2 distance (68.0), which is far worse than using the k-NN classifier (55.7, see Table 2). 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 [46] 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 algorithms which do allow generalization to new classes. First, we consider two other supervised metric learning approaches, NCM with FDA (which leads to 50.2) and ridge-regression (which leads to 54.6). We observe that NCMML outperforms both methods significantly. Sec-

Projection dim.	4K dimensional features						Full	64K dimensional features				
	32	64	128	256	512	1024		128	256	512	Full	
SVM baseline							38.2					28.0
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		31.7	31.0	30.7		
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					63.2
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		32.2	30.1	29.2		

Table 3: Flat top-5 error of k-NN and NCM classifiers, as well as baselines, using the 4K and 64K dimensional features, for various projection dimensions, and comparison to related methods, see text for details.

ond, 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-dimensional projections.

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.

5.3.1 Illustration of metric learned by NCMML.

In Figure 7 we illustrate the difference between the ℓ_2 and the Mahalanobis metric induced by a learned projection from 64K to 512 dimensions. For three 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 \mathbf{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. (9). 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


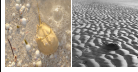




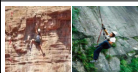


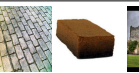























	 horseshoe crab 0.99%	 African elephant 0.99%	 mongoose 0.94%	 Indian elephant 0.88%	 dingo 0.87%	L2
Cliff dwelling L2 11.0% - Mah. 99.9%	 cliff 0.07%	 dam 0.00%	 stone wall 0.00%	 brick 0.00%	 castle 0.00%	Mah.
	 shopping cart 1.07%	 unicycle 0.84%	 covered wagon 0.83%	 garbage truck 0.79%	 forklift 0.78%	L2
Gondola L2 4.4% - Mah. 99.7%	 dock 0.11%	 canoe 0.03%	 fishing rod 0.01%	 bridge 0.01%	 boathouse 0.01%	Mah.
	 crane 0.87%	 stupa 0.83%	 roller coaster 0.79%	 bell cote 0.78%	 flagpole 0.75%	L2
Palm L2 6.4% - Mah. 98.1%	 cabbage tree 0.81%	 pine 0.30%	 pandanus 0.14%	 iron tree 0.07%	 logwood 0.06%	Mah.

Fig. 7: 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.

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.

5.3.2 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 8, we show the performance of using the NCMC classifier only at test time with $k = [2, \dots, 30]$, while using a metric obtained by the NCM objective ($k = 1$). This method is denoted as NCMC-test. In Table 4, we show the performance of the NCMC classifier, trained with the NCMC objective, using the 4K features. In the same table we compare the results 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 dimensionalities. When learning 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, we obtain a performance of

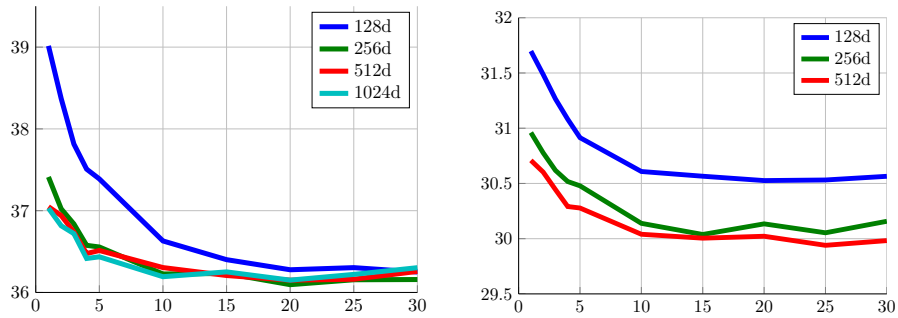


Fig. 8: The flat top-5 error of the NCMC-test classifier, which at test time uses $k > 1$ on a metric obtained with $k = 1$, for the 4K features (*left*) and the 64k (*right*) features.

Proj. Dim.	NCM	NCMC-test (k)	NCMC		
			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

Table 4: The flat top-5 error of the NCMC classifier using the 4K features, compared to the NCM baseline and the best NCMC-test classifier (with k in brackets).

34.6 on the top-5 error, using $k = 10$ centroids. That 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 3.

For the 64K features learning for the NCMC objective (with, $k = 10$ and $d = 512$) improves the performance to 29.4, about 1.3 points over the NCM classifier.

5.4 Generalizing to new classes with few samples

Given the encouraging classification accuracy of the NCM classifier observed above—and its superior efficiency as compared to the k -NN classifier—we now explore its ability to generalize to novel classes. 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 use approximately 1M images corresponding to 800 random classes to learn metrics, and evaluate the generalization performance on 200 held-

	4K dimensional features						64K dimensional features							
	SVM	k-NN			NCM			SVM	NCM					
Projection dim.	Full	128	256	ℓ_2	128	256	512	1024	ℓ_2	Full	128	256	512	ℓ_2
Trained on all	37.6	39.0	38.4		38.6	36.8	36.4	36.5		27.7	31.7	30.8	30.6	
Trained on 800		42.2	42.4	54.2	42.5	40.4	39.9	39.6	66.6		39.3	37.8	37.8	61.9

Table 5: Flat top-5 error for 1,000-way classification among test images of 200 classes not used for metric learning, and control setting with metric learning using all classes.

Method	4K dimensional features						64K dimensional features				Previous Results				
	NCM				SVM		NCM				SVM		[9]	[39]	[35]
Proj. dim.	128	256	512	1024	Full	Full	128	256	512	Full	Full	21K	128K	128K	
Top-1 err.	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	80.9	80.8
Top-5 err.	80.7	78.7	78.6	78.6	89.0	72.4	71.7	70.5	70.1	85.4	60.9				

Table 6: Flat error rate of the NCM classifier on the ImageNet-10K dataset, using metrics learned on the ILSVRC’10 dataset, with comparison to the baseline SVM, the NCM using ℓ_2 distance (denoted as full), and previously reported SVM results [9, 39, 35] and the Deep Learning framework of [26].

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. The early stopping strategy uses the validation set of the 200 unseen classes. Performance among test images of the 800 train classes changes only marginally and would obscure the changes among the test images of the 200 held-out classes.

In Table 5 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 39.6 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.

To further demonstrate the generalization ability of the NCM classifier using learned metrics, we also compare it against the SVM baseline 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 6 show that even in this extremely challenging setting the NCM classifier performs remarkably well compared to earlier mentioned SVM-based results of [9, 39, 35] and our baseline, all of which require training 10K classifiers. We note that, to the best of our knowledge,

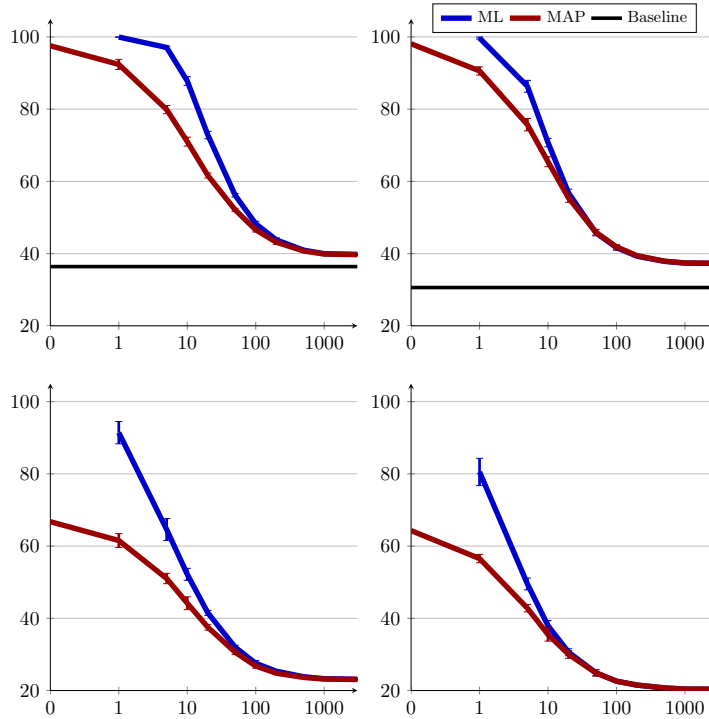


Fig. 9: Flat top-5 error of NCM as a function of the number of images used to compute the means for test classes. We compare the ML (*blue*) and MAP (*red*) mean estimates, for the 4K (*left*) and 64K (*right*) features. We show the results for 1,000-way (*top*), including baseline when trained on all classes in *black*, and 200-way classification (*bottom*).

our baseline results exceed the previously known state-of-the-art [35, 26] on this dataset. Training our SVM baseline system took 9 and 280 CPU 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 baseline, without counting the time to learn the 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 [37], we also include results of a zero-shot learning experiment, where we use the ImageNet hierarchy to estimate the mean of novel classes from the means of related training classes, see Section 3.6 for details.

In Figure 9 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, both in a 1,000-way classification as above, and in a 200-way classification as in [37]. 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 also show that the prior leads to zero-shot performance of 66.5 (4K features) and 64.0 (64K features), in the 200-way classification setting. These results are comparable to the result of 65.2 reported in [37], even though a different set of 200 test-classes were used. Note that they also used different features, however their baseline performance of 37.6 top-5 error is comparable to our 4K features (38.2).

More importantly, we 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 leads to a significant error reduction in the regime where ten images or less are available.

5.5 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. In this case the class mean simplifies to the query. We propose to use a metric learned for our NCM classifier on an auxiliary supervised dataset to retrieve the most similar images for a given query.

Using classifiers to learn a metric for image retrieval was recently considered also in [16]. They found the Joint Subspace and Classifier Learning (JSCL) method to be most effective. This basically amounts to jointly learning a set of classifiers and a projection matrix W using the WSABIE scoring function, Eq. (15), and minimizing the hinge-loss on class labels. After training, the classifiers are discarded and only the learned projection matrix W is used to compute distances between query and database images.

For this experiment we use the same public benchmarks as in [16]. First, the INRIA Holidays data set introduced by [20] consists of 1,491 images of 500 scenes and objects. In the standard evaluation protocol, one image per scene / object is used as query to search withing the remaining images. The accuracy is measured as the mean average precision over the 500 queries (mAP). Second, the University of Kentucky Benchmark dataset (UKB) introduced by [32], which contains of 4 images for each of the 2,550 objects (10,200 images). We follow the standard evaluation protocol, where each image is used as query to search in the database. The performance is measured by $4 \times \text{recall}@4$ averaged over all queries, hence the maximal score is 4. For both datasets we extract the 4K image features also used in our earlier experiments, which are the same ones as those used in [16]. To compute the

Dim	INRIA Holidays dataset no projection: 77.4%				UKB dataset no projection: 3.19		
	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

Table 7: Instance level image retrieval accuracy on the Holidays dataset, and the UKB dataset. NCMML is compared to a PCA baseline and the JSCL method [16].

distance between two images, we use the cosine-distance, *i.e.* the dot-product on ℓ_2 -normalized vectors.

In analogy to [16] we use NCMML to train a metric from the ILSVRC’10 data set, and do the early stopping based on retrieval performance. To avoid tuning on the test data the cross-validation is performed on the other dataset, *i.e.* when testing on UKB and we regularize based on Holidays and vice-versa. In Table 7 we compare the performance of the NCM based metric with that of JSCL, and also include a baseline PCA method and the performance using the high-dimensional descriptors without any projection. Finally, for the Holidays dataset we included the NCM metric while using early-stopping based on classification performance on the ILSVRC’10 validation data set (NCM-class).

From the results we observe that the NCM metric yields similar performance gains as the JSCL method on both datasets. In both cases a projection to only 128 dimensions yields an equal or better retrieval performance as using the original 4K dimensional features. 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, employing class level labels, and outperform the unsupervised PCA projection.

Note that it is crucial to use retrieval performance for early stopping; without it the results (see NCM*) are in fact worse than the original descriptors, and comparable to using PCA. Thus, the classification objective determines a good “path” through the space of projection matrices, yet it is crucial to regularize for retrieval performance, where the selected 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 needed for good classification. This observation 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 chapter we have considered distance-based classifiers for large-scale image classification. The advantage of distance-based classifiers is that new data (possibly of new classes) can be integrated at a negligible cost. This advantageous property is not shared by the one-vs-rest SVM approach that is used in most current state-of-the-art approaches, but is essential when dealing with real-life open-ended datasets where new images and classes are continuously added over time.

Since the performance of distance-based classifiers is heavily dependent on the used metric, we employ supervised metric learning techniques to improve classification performance. For k-NN classifiers we rely on the large margin nearest neighbor (LMNN) framework, and consider several variants to select the set of target neighbors. For the NCM classifier we have introduced a new metric learning technique, which we coined NCMML. It is based on maximizing the log-likelihood of correct prediction using a soft-min over the distances to class centers to define the class probabilities of a sample. Moreover, we introduced the NCMC classifier, a non-linear extension of the NCM classifier, that uses multiple centroids to represent each class. The used number of centroids offers a complexity trade-off from the linear NCM classifier to the non-linear and non-parametric k-NN classifier where all samples are used as a class centroids.

We have experimentally compared the k-NN and NCM classifiers to a strong baseline, the one-vs-rest SVM approach. Using a high-dimensional Fisher vector image representation our baseline attains current state-of-the-art results. Surprisingly we found that the NCM classifier outperforms the more flexible k-NN approach. Moreover, the performance of the NCM classifier is comparable to that of SVM baseline (slightly better with 4K dimensional features, but somewhat worse using the 64K dimensional features), while projecting the data to as few as 256 dimensions. Furthermore, we find that using multiple centroids per class improves the performance of the NCM classifier.

Our learned metrics generalize well to unseen classes, as shown by the experiments where the metric is learned on a subset of 800 classes and evaluated on the 200 held out classes. In this case we observe only a modest drop in performance, in spite of the fact that for the held-out classes “training” consists only in computing the mean vector of each class. These results are further corroborated by our experiments on the ImageNet-10K dataset, where we obtain competitive performance at a negligible cost compared to the feature extraction process. We only need to compute the class means, as opposed to training 10,000 binary one-vs-rest SVM classifiers, which represents roughly a 8,500 fold speedup.

In addition, we have shown that our NCM classifiers can be used in a zero-shot setting where no training images are available for novel classes, and that the zero-shot prior significantly improves performance when combined with a class mean estimated from a limited amount of training images.

Finally we have shown that NCM provides a unified way to treat classification and retrieval problems, as query-by-example image retrieval can be seen as a classification problem where only a single positive sample per class is provided. We

have evaluated the NCM metric for image retrieval and found performance that is comparable to previous published metric learning approaches.

References

1. 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*, 2010.
2. S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2011.
3. L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.
4. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
5. J. Chai, H. Liua, B. Chenb, and Z. Baoa. Large margin nearest local mean classifier. *Signal Processing*, 2010.
6. G. Checkik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.
7. S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders. XRCE’s participation to ImageEval. In *ImageEval Workshop at CVIR*, 2007.
8. 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.
9. J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.
10. 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.
11. L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. PAMI*, 28(4):594–611, 2006.
12. T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.
13. 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*, 1994.
14. A. Globerson and S. Roweis. Metric learning by collapsing classes. In *NIPS*, 2006.
15. J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *NIPS*, 2005.
16. A. Gordo, J. Rodríguez, F. Perronnin, and E. Valveny. Leveraging category-level labels for instance-level image retrieval. In *CVPR*, 2012.
17. R. Gray and D. Neuhoff. Quantization. *IEEE Trans. Information Theory*, 44(6):2325–2383, 1998.
18. M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.
19. M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *ICCV*, 2009.
20. H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.
21. H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 2011.
22. 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.
23. M. Köstinger, M. Hirzer, P. Wohlhart, P. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *CVPR*, 2012.
24. C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

25. H. Larochelle, D. Erhan, and Y. Bengio. Zero-data learning of new tasks. In *AAAI Conference on Artificial Intelligence*, 2008.
26. 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.
27. 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.
28. D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
29. A. Lucchi and J. Weston. Joint image and word sense discrimination for image retrieval. In *ECCV*, 2012.
30. 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.
31. T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Trans. PAMI*, 2012 Submitted.
32. D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.
33. E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *CVPR*, 2007.
34. S. Parameswaran and K.Q. Weinberger. Large margin multi-task metric learning. In *NIPS*, 2010.
35. F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. In *CVPR*, 2012.
36. F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.
37. M. Rohrbach, M. Stark, and B. Schiele. Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In *CVPR*, 2011.
38. K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.
39. J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.
40. 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.
41. C. Veenman and D. Tax. LESS: a model-based classifier for sparse subspaces. *IEEE Trans. PAMI*, 27(9):1496–1500, 2005.
42. A. R. Webb. *Statistical pattern recognition*. Wiley, New-York, NY, USA, 2002.
43. K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006.
44. K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.
45. K.Q. Weinberger and O. Chapelle. Large margin taxonomy embedding for document categorization. In *NIPS*, 2009.
46. J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.
47. J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 73(2):213–238, 2007.
48. 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.

Index

Fisher discriminant analysis, 10

Image classification, 2, 4

Image retrieval, 31

k-Nearest Neighbors, 17

Large-scale classification, 4

Linear classification, 10

Mahalanobis distance, 8

Metric learning, 5, 8, 17

Multi-class logistic regression, 8

Nearest class mean classifier, 7

Regularized linear least-squares regression, 11

Ridge-regression, 11

Transfer learning, 6, 16, 28

Zero-shot classification, 16, 30