Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost

Thomas Mensink[1,2], Jakob Verbeek[1], Florent Perronnin[2], and Gabriela Csurka[2]

[1] LEAR, INRIA Grenoble, 655 Avenue de l'Europe, 38330 Montbonnot, France
email: firstname.lastname@inria.fr
[2] TVPA, Xerox Research Centre Europe, 6 chemin de Maupertuis, 38240 Meylan, France
email: firstname.lastname@xrce.xerox.com

Abstract. We are interested in large-scale image classification and especially in the setting where images corresponding to new or existing classes are continuously added to the training set. Our goal is to devise classifiers which can incorporate such images and classes on-the-fly at (near) zero cost. We cast this problem into one of learning a metric which is shared across all classes and explore k-nearest neighbor (k-NN) and nearest class mean (NCM) classifiers. We learn metrics on the ImageNet 2010 challenge data set, which contains more than 1.2M training images of 1K classes. Surprisingly, the NCM classifier compares favorably to the more flexible k-NN classifier, and has comparable performance to linear SVMs. We also study the generalization performance, among others by using the learned metric on the ImageNet-10K dataset, and we obtain competitive performance. Finally, we explore zero-shot classification, and show how the zero-shot model can be combined very effectively with small training datasets.

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. Only a small fraction of this data is consistently annotated and thus scalable methods are needed for annotation and retrieval to efficiently access this huge volume of data. 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, see among others [1,2,3,4,5,6,7,8,9,10]. The introduction of the ImageNet dataset [1], 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 paper we focus on the problem of large-scale image annotation, where the goal is to assign automatically a set of relevant labels to an image, *e.g.* names of objects appearing in the image. The predominant approach to this problem is to treat it as a classification problem. To ensure scalability, often linear classifiers such as SVMs are used [8,10], sometimes in combination with dimension reduction techniques to speed-up the classification [7]. Recently, impressive results have been reported on 10,000 or more classes [3,7,8,11]. A drawback of these methods, however, is that when images of new categories become available, new classifiers have to be trained at a relatively high computational cost.

Many real-life large-scale datasets are open-ended and dynamic: new classes appear over time and new photos/videos are continuously added to existing classes. Therefore, the main objective of our work is to propose and study approaches which enable the addition of new classes and new images to existing classes at (near) zero cost. We explore two techniques to do so. The first is k-NN classification, where images of new classes are added in the database, and can be used for classification without further processing. This is a highly non-linear and non-parametric classifier that has shown competitive performance for image annotation [3,7,12]. Its main drawback is that the nearest neighbor search for classification is computationally demanding for large and high-dimensional datasets. The second is the nearest class mean classifier (NCM), where classes are represented by their mean feature vector [13,14,15]. The cost of computing the mean can be neglected with respect to the cost of feature extraction and this operation does not require accessing images of other classes. Contrary to the k-NN classifier, this is a linear classifier which leads to efficient classification. The complete distribution of the training data of a class is, however, only characterized by its mean and it is therefore unclear whether this is sufficient for competitive performance.

The success of both methods critically depends on the metric which is used to compute the distance between an image and other images (for k-NN) or class means (for NCM), see Figure 1 for an illustration. Metric learning has received much attention in the machine learning and computer vision communities recently, and has been shown to significantly increase the performance of distance-based classifiers. Therefore, we cast our classifier learning problem as one of learning a metric which is shared across all classes. For k-NN classification, we use the Large Margin Nearest Neighbor (LMNN) framework [16] and propose a variation that significantly improves its performance. For the NCM classifier, we propose a novel metric learning algorithm based on multi-class logistic discrimination. A sample from a given class is enforced to be closer to its class mean than to any other class mean in the projected space. To apply these metric learning techniques on large-scale datasets, we employ stochastic gradient descend (SGD) algorithms, which access only a small fraction of the training data at each iteration [17]. To allow metric learning on high-dimensional image features of large scale datasets that are too large to fit in memory, we in addition use product quantization [18], a data compression technique that was recently used with success for large-scale image retrieval [6] and classifier training [8].

We report experiments on the ImageNet Large Scale Visual Recognition Challenge 2010 (ILSVRC'10) dataset, which consists of 1.2M train images of 1,000 classes. As a baseline approach, we use the setup of the winning entry in the 2011 edition of the challenge [8]: Fisher vector image representations [19] are used to describe images and one-vs-rest linear SVM classifiers are learned independently for each class. Surprisingly, we find that the NCM classifier outperforms the more flexible k-NN classifier. Moreover, the NCM classifier performs on par with the SVM baseline. We also consider the generalization performance to new classes. In a first experiment, we train the metric on a subset of classes of ILSVRC'10 and include the held-out classes at test time. We only observe a small drop in performance compared to the experiment where the metric is learned with all classes. In a second experiment, we train the metric on ILSVRC'10 and apply it to a larger set of 10K ImageNet classes. Once the metric is

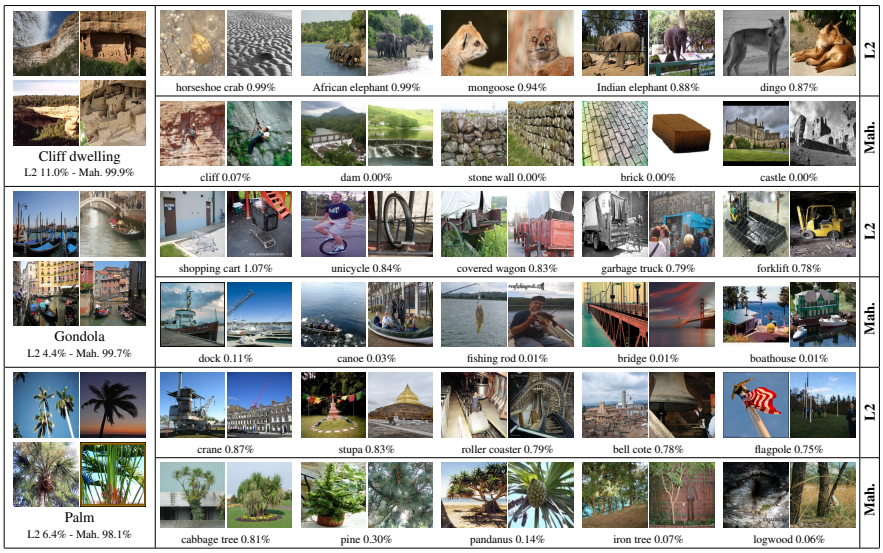

Fig. 1. The nearest classes for three reference classes using the the ℓ_2 and Mahalanobis metric learned for the NCM classifier. Class probabilities are given for a simulated image signature that equals the mean of the reference class, see Section 4.3 for details.

learned, we can learn the 10K classifiers (*i.e.* compute their means) on 64K dimensional features in less than an hour on a single CPU, while learning one-vs-rest linear SVMs on the same data takes on the order of 280 CPU days. Finally, we explore a zero-shot setting where the class mean of novel classes are estimated based on related classes in the ImageNet hierarchy. We show that the zero-shot class mean can be effectively combined with the empirical mean of a small number of training images. This provides an approach that smoothly transitions from settings without training data to ones with abundant training data.

The rest of the paper is organized as follows. In Section 2 we discuss a selection of related work which is most relevant to this paper. In Section 3 we present the metric learning techniques we explored for the k-NN and NCM classifiers. We then present extensive experimental results in Section 4, followed by our conclusions in Section 5.

2 Related Work

In this section we discuss some of the most relevant work on large-scale image annotation, metric learning, and transfer learning.

Large-scale image annotation. The ImageNet dataset [1] has been a catalyst for research on large-scale image annotation. The current state-of-the-art [8,10] uses efficient linear SVM classifiers trained in a one-vs-rest manner in combination with high-dimensional bag-of-word [20] or Fisher vector representations [19]. Besides one-vs-rest

training, large-scale ranking-based formulations have also been explored in [7]. Interestingly, this 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 reduces the cost of classifier evaluation at test time. The proposed NCM approach also learns low-dimensional projection matrices but the weight vectors are constrained to be the class means. This allows the efficient addition of novel classes.

In [3,7] k-NN classifiers were found to be competitive with linear SVM classifiers in a very large-scale setting involving 10,000 or more classes. The drawback of k-NN classifiers, however, is that they are expensive in storage and computation, since in principle all training data needs to be kept in memory and accessed to classify new images. 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 [6] 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-word and Fisher vector image representations in the context of image retrieval [9] and classifier training [8]. We also exploit PQ in our work to compress high-dimensional image signatures when learning our metrics.

Metric learning. Here, we only discuss methods that learn metrics for image classification problems. Other methods aim at learning metrics for verification problems and essentially learn binary classifiers that threshold the learned distance to decide whether two images belong to the same class or not, see *e.g.* [21].

Among those methods that learn metrics for classification, the Large Margin Nearest Neighbor (LMNN) approach of [16] is specifically designed to support k-NN classification. It tries to ensure that for each image its 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. In [16] the set of target neighbors is chosen and fixed using the ℓ_2 metric in the original space, that can be problematic as this distance is far from being ideal, see *e.g.* Figure 1. Therefore, we propose two variants of LMNN that avoid using such a pre-defined set of target neighbors, both leading to significant improvements.

The large margin nearest local mean classifier [22] assigns a test image to a class based on its distance to the mean of its nearest neighbors in each class. This method was reported to outperform LMNN but requires computing all pairwise distances between training instances and therefore does not scale well to large datasets. Similarly, TagProp [12] suffers from the same problem. It consists in assigning weights to training samples based on their distance to the test instance and in computing the class prediction by the total weight of samples of each class in a neighborhood. Because of their poor scaling properties, we do not consider these methods in our experiments.

Closely related to our metric learning approach for the NCM classifier is the LESS model of [14]. 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. The method of [15] is also related to our method since they use a NCM classifier and an ℓ_2 distance in a subspace that is orthogonal to the subspace with maximum within-class variance. However, their technique involves computing the first eigenvectors of the within-class covariance matrix, which has a computational cost between $O(D^2)$ and $O(D^3)$, which again is undesirable for high-dimensional feature vectors. Moreover, this metric is heuristically obtained, rather than directly optimized for maximum classification performance.

Transfer learning. The term transfer learning is used to refer to methods that share information across classes during learning. Examples include the use of part-based or attribute class representations. Part-based object recognition models [23] 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.* animal) by a combination of attributes (*e.g.* yellow, stripes, 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 [25]. Zero-shot learning [26] 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 seen classes. In [5] various transfer learning methods were evaluated in a large-scale setting using the ILSVRC'10 dataset. They found transfer learning methods to have little added value when training images are available for all classes. In contrast, transfer learning was found to be effective in a zero-shot learning setting, where classifiers were trained for 800 classes, and performance was tested in a 200-way classification across the held-out classes.

In this paper we also aim at transfer learning, in the sense that we allow only a trivial amount of processing on the data of new classes (storing in a database, or averaging), and rely on a metric that was trained on other classes to recognize the new ones. In contrast to most work 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 [5] 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 Learning Metrics to Generalize to New Categories

In this section we discuss metric learning for the k-NN and NCM classifiers. We learn Mahalanobis distances of the form $(x - x')^\top M(x - x')$ to improve classification accuracy, where $x \in \mathbb{R}^D$. 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 ℓ_2 distance after projection:

$$\| x - x' \|_W^2 = (x - x')^\top W^\top W(x - x') = \| Wx - Wx' \|^2 . \quad (1)$$

3.1 Metric Learning for k-NN Classification

K-NN classification is essentially a ranking problem, which is reflected in the metric learning approach of LMNN [16]. Their learning objective is based on triplets of images, where the distance between a query image q and a target image p of the same class should be smaller than the distance to a negative image n of a different class. The 0/1-loss for such a triplet is upper-bounded by the hinge-loss on the distance difference:

$$L_{qpn} = \big[1 + \| x_q - x_p \|^2_W - \| x_q - x_n \|^2_W \big]_+, \tag{2}$$

which is zero if the negative image n is at least one distance unit farther from the query q than the positive image p, and positive otherwise. The sum of the per-triplet loss is the final learning criterion:

$$L = \sum_{q=1}^{N} \sum_{p \in P_q} \sum_{n \in N_q} L_{qpn}, \tag{3}$$

where P_q and N_q denote the set of positive and negative images for a query image x_q. The sub-gradient of the loss is obtained as:

$$\nabla_W L = \sum_{q=1}^{N} \sum_{p \in P_q} \sum_{n \in N_q} \nabla_W L_{qpn}, \tag{4}$$

$$\nabla_W L_{qpn} = [\![L_{qpn} > 0]\!] 2W \Big((x_q - x_p)(x_q - x_p)^\top - (x_q - x_n)(x_q - x_n)^\top \Big), \tag{5}$$

where we use Iversons bracket notation $[\![\cdot]\!]$ that equals one if its argument is true, and zero otherwise.

In LMNN the set of targets P_q for a query q is set to the query's k nearest neighbors from the same class, using the ℓ_2 distance. The rationale is the following one: if we can ensure that these targets are closer than the instances of the other classes, then the k-NN classification will succeed. In practice, however, it is not always possible to achieve this situation with a given set of target neighbors, since it implicitly assumes that the ℓ_2 distance in the original space is a good similarity measure. Therefore, we consider two alternatives to using a fixed set of target neighbors:

- First, we can set P_q to contain all images with the same class label as q. This is similar to [2] where the same type of loss was used to learn image similarity defined as the scalar product between images after a learned linear projection.
- Second, we can define P_q dynamically to contain the k images of the same class that are closest to q using the current metric. This option corresponds to minimizing the loss function also with respect to the choice of P_q. Hence, different target neighbors can be selected, that are closer than the original ones according to the current metric. A similar approach has been proposed in [16], where every T iterations P_q is redefined using target neighbors according to the current metric.

Since the number of possible triplets tends to be large for any of these strategies[3], at each SGD iteration we generate triplets from a limited set of $m \ll N$ images.

[3] Even if we disregard the set of target neighbors, each query image is paired with all images of other classes. Thus, if we have N training images evenly distributed among C classes, this

3.2 Metric Learning for the Nearest Class Mean Classifier

We formulate the NCM classifier using multi-class logistic regression and define the probability for a class c given an image feature vector x as:

$$p(c|x) = \frac{\exp - \parallel \mu_c - x \parallel^2_W}{\sum_{c'=1}^{C} \exp - \parallel \mu_{c'} - x \parallel^2_W}, \tag{6}$$

where μ_c is the mean for class $c \in \{1, \ldots, C\}$. Our objective is to minimize the negative log-likelihood of the ground-truth class labels $y_i \in \{1, \ldots, C\}$ of the training images:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \ln p(y_i|x_i). \tag{7}$$

The gradient of this objective function is easily verified to be:

$$\nabla_W \mathcal{L} = \frac{2}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} \Big([\![y_i = c]\!] - p(c|x) \Big) W(\mu_c - x_i)(\mu_c - x_i)^\top. \tag{8}$$

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.

Note that the NCM classifier is linear in x since we assign an image x to the class c^* with minimum distance:

$$c^* = \arg\min_c \left\{ \parallel x - \mu_c \parallel^2_W \right\} = \arg\min_c \left\{ \parallel W\mu_c \parallel^2 - 2\mu_c^\top (W^\top W) x \right\}. \tag{9}$$

This observation shows that our classifier can be related to the WSABIE method of [7]. In the latter paper, a class c is scored using $f_c(x) = v_c^\top W x$, where W projects the image into a lower-dimensional space, and v_c is a class specific weight vector learned from data. However, we enforce that $v_c = W\mu_c$, which allows us to add new classes without the need to learn v_c from labeled data and therefore at almost zero cost.

Note that our classifier can also be related to the solution of ridge-regression, or regularized linear least-squares regression. Let $f_c(x_i) = b_c + w_c^\top x_i$ and let $y_{ic} = 1$, if x_i belongs to class c, and $y_{ic} = 0$ otherwise. The ridge-regression loss function, $\frac{1}{N} \sum_i (f_c(x_i) - y_{ic})^2 + \lambda \parallel w_c \parallel^2$, can be minimized in closed form and leads to $f_c(x) = \frac{n_c}{N} \big(1 + \mu_c^\top (\Sigma + \lambda I)^{-1} x\big)$, where Σ is the (class-independent) data covariance matrix, and n_c denotes the number of images in class c. If n_c is equal for all classes, then the score function is equivalent to $\mu_c^\top (\Sigma + \lambda I)^{-1} x$ up to additive and multiplicative constants. This is similar to our NCM classifier if we set W such that $W^\top W = (\Sigma + \lambda I)^{-1}$, see Eq. (9). Moreover, just like the NCM classifier, the ridge-regression function for a new class can also be found from the class mean μ_c and count n_c, once the data covariance matrix has been estimated.

---

would yield already $N(N - \frac{N}{C})$ pairs. For ILSVRC'10, we have $N \approx 10^6$ and $C = 10^3$ which leads to roughly 10^{12} pairs, and to even more triplets.

4 Experimental Evaluation

In this section we first describe the dataset and evaluation measures used in our experiments. Then, we present our results for metric learning using k-NN classification in Section 4.2, and those using NCM classification in Section 4.3. Finally, in Section 4.4, we present results for transfer learning, where we apply metrics trained on some classes to novel ones, and assess performance as a function of the number of training images.

4.1 Experimental Setup and Baseline Approach

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

Features. We represent each image, with a Fisher vector (FV) [19] computed over densely extracted SIFT descriptors and local color features, both projected with PCA to 64 dimensions. FVs are extracted and normalized separately for both channels and then combined by concatenating the two feature vectors. We do not make use of spatial pyramids. In our experiments we use FVs extracted using a vocabulary of either 16 or 256 Gaussians. For 16 Gaussians, this leads to a 4K dimensional feature vector, which requires about 20GB for the 1.2M training set (using 4-byte floating point arithmetic). This fits into the RAM of our 32GB servers.

For 256 Gaussians, the FVs are 16 times larger, *i.e*. 64K dimensional, which would require 320GB of memory. Hence, we compress the feature vectors using product quantization [18]. 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. 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. 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. In tables, we highlight the best result per row in bold, and do so for each feature set if several are used. Additionally, the baseline performance is also highlighted if it is best.

Baseline approach. For our baseline, we follow the state-of-the-art approach of [8] and learn 1,000 one-vs-rest SVMs with SGD. The results of the baseline can be found in Table 2. We see that the 64K dimensional features lead to significantly better results than the 4K ones, despite the lossy PQ compression. The performance using the 64K features is slightly better than the ILSVRC'10 challenge winners [10] (28.0 vs. 28.2 flat top-5 error), and very close to the results of [8] (25.7 flat top-5 error). Note that there a much higher dimensional representation was used of more than 1M features.

Table 1. k-NN classification performance with 4K dimensional features. For all methods, except those indicated by 'Full', the data is projected to a 128 dimensional space.

		k-NN classifiers						
	SVM	ℓ_2	ℓ_2	LMNN		All	Dynamic	
	Full	Full	+ PCA	10	20		10	20
Flat top-1 error	**60.2**	75.0	76.3	72.9	72.8	67.9	**65.1**	66.0
Flat top-5 error	**38.2**	55.9	57.3	50.6	50.4	44.2	**39.8**	40.7

SGD training and early stopping. To learn the projection matrix W, we use SGD training, sampling at each iteration a fixed number of training images to estimate the gradient. Following [27], we use a fixed learning rate and use the projection dimension $d \leq D$ as well as the number of iterations as an implicit form of regularization. We run SGD for 750K-4M iterations, and the performance is validated every 10K (NCM) or 50K (k-NN) iterations. The metric which yields the lowest top-5 error is selected. 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 on the provided 50K validation set.

4.2 k-NN Metric Learning Results

We start with an assessment of k-NN classifiers using metrics learned with the methods described in Section 3.1. Given the cost of k-NN classifiers, we focus our experiments on the 4K features. We consider the impact of the different choices for the set of target images P_q as described in Section 3.1, and the influence of the projection dimensionality. We initialize W as a PCA projection, the optimal k is typically 100 or 250.

Target selection for k-NN metric learning. In the first experiment we compare the three different options of Section 3.1 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 1 show that all methods lead to metrics that are better than the ℓ_2 metric in the original space, or after a PCA projection to 128 dimensions. Furthermore, we can improve over LMNN by using all within-class images as targets, or even further by using dynamic targets. The success of the dynamic target selection can be explained by the fact that among the three alternatives, it is the most closely related to the k-NN classifier objective. The best performance on the flat top-5 error of 39.8 using 10 dynamic targets is, however, slightly worse than the 38.2 error rate of the SVM baseline.

Impact of projection dimension on k-NN classification. Next, we evaluate the influence of the projection dimensionality, by varying it 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 2 we see that a projection to 256 dimensions yields the lowest error of 39.0, still somewhat inferior of those of the SVM baseline.

Table 2. Performance of k-NN and NCM classifiers, as well as baselines, using the 4K and 64K dimensional features, for various projection dimensions, see text for details.

	4K dimensional features							64K dimensional features			
Projection dim.	32	64	128	256	512	1024	Full	128	256	512	Full
SVM baseline							**38.2**				**28.0**
k-NN, dynamic 10	47.2	42.2	39.8	**39.0**	39.1	40.4					
NCM, learned metric	49.1	42.7	39.0	37.4	**37.0**	**37.0**		31.7	31.0	**30.7**	
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**				
PCA+Ridge-regress.	86.3	80.3	73.9	68.1	62.8	58.9	**54.6**				
WSABIE [7]	51.9	45.1	41.2	39.4	38.7	**38.5**		32.2	30.1	**29.2**	

4.3 Nearest Class Mean Classification Results

We now consider the performance of NCM classifiers and the related methods described in Section 3.2. In Table 2 we show the results for various projection dimensionalities.

We first consider the results for the 4K dimensional features. Our first, unexpected, observation is that our NCM classifier (37.0) outperforms the more flexible k-NN classifier (39.0), and even slightly outperforms the SVM baseline (38.2) when projecting to 256 dimensions or more. Interestingly, using the ℓ_2 instead of a learned metric, the situation is reversed and the k-NN classifier is better (55.9, see Table 1) than the NCM classifier (68.0). WSABIE scores slightly worse (38.5) than the baseline and our NCM classifier, and does not generalize to new classes without retraining. Like the NCM classifier, ridge-regression does allow generalization to new classes, but leads to significantly worse results (54.6) and pre-processing the data with PCA further degrades its performance.

We also consider two 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 the metric generated by the inverse of the regularized covariance matrix, 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 as compared to a learned metric. Second, the feature whitening implemented by the inverse covariance metric, leads to results that are better than using the ℓ_2 metric, and also substantially better than ridge-regression. 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 (30.7) are somewhat worse than the SVM baseline (28.0); again the learned metric is better than using ℓ_2 (63.2). WSABIE obtains an error of 29.2, in between the SVM and NCM.

In Figure 1 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 probabilities on the reference class and its five neighbor classes according to Eq. (6). The feature vector x is set as the mean of the reference class, *i.e.* a simulated

Table 3. Performance of 1,000-way classification among test images of 200 classes not used for metric learning, and control setting with metric learning using all classes. Left column denotes the number of training classes used, and "plain" denotes k-NN or NCM using the ℓ_2 distance.

	4K dimensional features									64K dimensional features				
	SVM	k-NN			NCM					SVM	NCM			
	Full	128	256	Full	128	256	512	1024	Full	Full	128	256	512	Full
Plain				54.2					66.6					61.9
1000	37.6	39.1	38.4		38.6	36.8	**36.4**	36.5		**27.7**	31.7	30.8	**30.6**	
800		42.2	42.4		42.5	40.4	39.9	**39.6**			39.3	37.7	**37.1**	

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. (7). This does not change the ordering of classes, but ensures that we can compare probabilities computed using both metrics. We find that, as expected, the learned metric has more semantically related class neighbors. 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.

4.4 Generalization to New Classes and Using 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 novel classes not seen during training. In this experiment we use approximately 1M images corresponding to 800 random classes to learn metrics, and evaluate the error of a 1,000-way classification across all classes. We report the error computed over the 30K images in the test set of the held-out 200 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 3 we show the performance of NCM and k-NN classifiers for several projection dimensions, and compare it to two control settings, training on all 1,000 classes, and using the ℓ_2 distance. The results show that both classifiers generalize remarkably well to new 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 it is still surprisingly good considering that training for the novel classes consists only in computing the mean.

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. This dataset, introduced in [3], consists of 4.5M training images of 10,184 classes, and a test set of another 4.5M images. We use projections learned on the ILSVRC'10 dataset, and only compute the means of the 10K classes. The results in Table 4 show that even in this extremely challenging setting the NCM classifier performs remarkably well

Table 4. Average per-class performance of the NCM classifier on the ImageNet-10K dataset, using metrics learned on the ILSVRC'10 dataset, and comparison to previously published results.

	4K dimensional features					64K dimensional features				21K	128K	128K
Proj. dim.	128	256	512	1024	SVM	128	256	512	SVM	[3]	[8]	[11]
Flat top-1	91.8	90.7	90.5	**90.4**	**86.0**	87.1	86.3	**86.1**	**78.1**	93.6	83.3	81.9
Flat top-5	80.7	78.9	**78.6**	**78.6**	**72.4**	71.7	70.5	**70.1**	**60.9**			

compared to [3,8,11] and our baseline, all of which require training 10K classifiers. We note that, to the best of our knowledge, our baseline results exceed the previous known state-of-the-art [3,8,11] 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 a function of the number of training images of novel classes. In our last experiment we consider the error as a function of the number of images that are used to compute the means of novel classes. 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. We follow ideas of [5] and estimate the mean of a novel class as the average of the means associated with all ancestor nodes in the ILSVRC'10 class hierarchy. The means of internal nodes are computed as the average of the means of all descendant training classes. If we view the estimation of a class mean as the estimation of the mean of a Gaussian distribution, then the sample average μ_s corresponds to the Maximum Likelihood (ML) estimate and the zero-shot estimate μ_z can be thought of as a prior. We can combine this prior with the ML estimate to obtain a maximum a-posteriori (MAP) estimate μ_p on the class mean. The MAP estimate of the mean of a Gaussian is obtained as the ML estimate weighted by the number n of images that were used to compute it, plus the prior mean which has a weight m determined on the validation set [28], *i.e.* $\mu_p = (n\mu_s + m\mu_z)/(n+m)$.

In Figure 2 we analyze the performance of the NCM classifier trained on the images of the same 800 classes used above, with a learned projection from 64K to 512 dimensions. We again report the error among test images of the held-out classes, both in a 1,000-way classification as above, and in a 200-way classification as in [5]. 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 64.3, which is comparable to the result of 65.2 reported in [5], even though they used a different set of 200 test classes. 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.

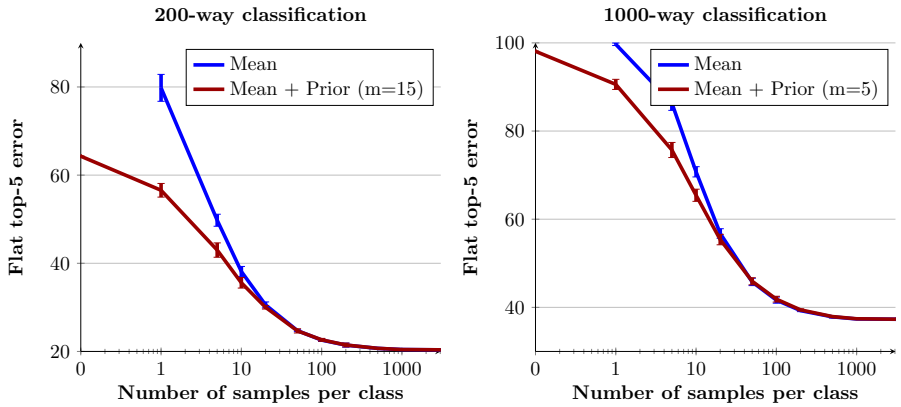

Fig. 2. Performance of NCM as a function of the number of images used to compute the means for classes not used during training, with and without the zero-shot prior. See text for details.

5 Conclusions

We have evaluated metric learning techniques to support k-NN and NCM classifiers, which can be applied on large scale dynamic and open-ended image datasets, and allow extensions at (near) zero cost to new classes not used for training. Surprisingly we found that the NCM classifier outperforms the more flexible k-NN approach. Moreover, using a learned metric, the performance of the NCM classifier is comparable to that of SVM classifiers, while projecting the data to only 256 dimensions. Our learned metrics generalize well to unseen classes, as shown in experiments where the metric is learned on a subset of the classes, and further corroborated by our experiments on the ImageNet-10K dataset. 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 model significantly boosts performance when combined with a class mean estimated from a limited amount of training images.

In the proposed NCM classifiers each class is represented only by its mean, while this has the advantage of obtaining fast linear classifiers at test time, this representation is also rather restrictive. In ongoing work we extend this class representation, for example by using a representation based on a set of class centroids, to obtain more flexible and non-linear classifiers.

Interestingly, query-by-example image retrieval can be seen as a classification problem where a single positive sample is provided. In such a case, the class mean simplifies to the query which shows that the proposed NCM provides a unified way to treat classification and retrieval problems. This is a direction that we will explore in future work.

References

1. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR. (2009)
2. Checkik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. Journal of Machine Learning Research **11** (2010) 1109–1135

3. Deng, J., Berg, A., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: ECCV. (2010)
4. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. In: CVPR. (2010)
5. Rohrbach, M., Stark, M., Schiele, B.: Evaluating knowledge transfer and zero-shot learning in a large-scale setting. In: CVPR. (2011)
6. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. PAMI **33** (2011) 117–128
7. Weston, J., Bengio, S., Usunier, N.: WSABIE: Scaling up to large vocabulary image annotation. In: IJCAI. (2011)
8. Sánchez, J., Perronnin, F.: High-dimensional signature compression for large-scale image classification. In: CVPR. (2011)
9. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE Trans. PAMI (2012) to appear.
10. Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L., Huang, T.: Large-scale image classification: Fast feature extraction and SVM training. In: CVPR. (2011)
11. Perronnin, F., Akata, Z., Harchaoui, Z., Schmid, C.: Towards good practice in large-scale learning for image classification. In: CVPR. (2012)
12. Guillaumin, M., Mensink, T., Verbeek, J., Schmid, C.: Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In: ICCV. (2009)
13. Webb, A.R.: Statistical pattern recognition. Wiley, New-York, NY, USA (2002)
14. Veenman, C., Tax, D.: LESS: a model-based classifier for sparse subspaces. IEEE Trans. PAMI **27** (2005) 1496–1500
15. Zhou, X., Zhang, X., Yan, Z., Chang, S.F., Hasegawa-Johnson, M., Huang, T.: SIFT-Bag kernel for video event analysis. In: ACM Multimedia. (2008)
16. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research **10** (2009) 207–244
17. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: COMPSTAT. (2010)
18. Gray, R., Neuhoff, D.: Quantization. IEEE Trans. Information Theory **44** (1998) 2325–2383
19. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher kernel for large-scale image classification. In: ECCV. (2010)
20. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: a comprehensive study. IJCV **73** (2007) 213–238
21. Nowak, E., Jurie, F.: Learning visual similarity measures for comparing never seen objects. In: CVPR. (2007)
22. Chai, J., Liua, H., Chenb, B., Baoa, Z.: Large margin nearest local mean classifier. Signal Processing **90** (2010) 236–248
23. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. PAMI **28** (2006) 594–611
24. Lampert, C., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: CVPR. (2009)
25. Tommasi, T., Caputo, B.: The more you know, the less you learn: from knowledge transfer to one-shot learning of object categories. In: BMVC. (2009)
26. Larochelle, H., Erhan, D., Bengio, Y.: Zero-data learning of new tasks. In: AAAI Conference on Artificial Intelligence. (2008)
27. Bai, B., Weston, J., Grangier, D., Collobert, R., Qi, Y., Sadamasa, K., Chapelle, O., Weinberger, K.: Learning to rank with (a lot of) word features. Information Retrieval – Special Issue on Learning to Rank **13** (2010) 291–314
28. Gauvain, J.L., Lee, C.H.: Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. IEEE Trans. Speech and Audio Proc. **2** (1994) 291–298