# Efficient Retrieval for Large Scale Metric Learning

Martin Köstinger, Peter M. Roth, and Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{*koestinger,pmroth,bischof*}*@icg.tugraz.at*

**Abstract.** In this paper, we address the problem of efficient k-NN classification. In particular, in the context of Mahalanobis metric learning. Mahalanobis metric learning recently demonstrated competitive results for a variety of tasks. However, such approaches have two main drawbacks. First, learning metrics requires often to solve complex and thus computationally very expensive optimization problems. Second, as the evaluation time linearly scales with the size of the data k-NN becomes cumbersome for large-scale problems or real-time applications with limited time budget. To overcome these problems, we propose a metric-based hashing strategy, allowing for both, efficient learning and evaluation. In particular, we adopt an efficient metric learning method for local sensitive hashing that recently demonstrated reasonable results for several large-scale benchmarks. In fact, if the intrinsic structure of the data is exploited by the metric in a meaningful way, using hashing we can compact the feature representation still obtaining competitive results. This leads to a drastically reduced evaluation effort. Results on a variety of challenging benchmarks with rather diverse nature demonstrate the power of our method. These include standard machine learning datasets as well as the challenging Public Figures Face Database. On the competitive machine learning benchmarks we obtain results comparable to the state-of-the-art Mahalanobis metric learning and hashing approaches. On the face benchmark we clearly outperform the state-of-the-art in Mahalanobis metric learning. In both cases, however, with drastically reduced evaluation effort.

## 1 Introduction

Among the various different classification schemes k-nearest neighbor (k-NN) based approaches using Mahalanobis metric learning have recently attracted a lot of interest in computer vision. Several powerful metric learning frameworks (*e.g.* [19, 20], [4], or [7]) have been proposed that study different loss functions or regularizations. Conceptually, these methods take advantage of prior information in form of labels over simpler though more general similarity measures. Significant improvements have been observed for tracking [17], image retrieval [10], face identification [12], clustering [23], or person re-identification [9].

The large-scale nature of computer vision applications poses several challenges and opportunities to the class of Mahalanobis metric learning algorithms. For instance we can learn a sophisticated distance metric that captures the structure of the dataset or learn multiple local metrics that better adapt to the intrinsic characteristics of the feature space. On larger datasets this usually leads to lower error rates [20]. In contrast, this is challenged by the computational burden in training and the needed label effort.

To reduce the required level of supervision, algorithms such as [4, 11] have been introduced that are able to learn from pairwise labels. Others tackle the problem of time complexity in learning by special optimization techniques [4, 20]. Nevertheless, one important aspect that is often neglected is the computational burden at test time as k-NN-search in high-dimensional spaces is cumbersome. For real-time applications with limited time budget this is even more critical; especially on larger datasets with tens of thousands of samples that have to be explored.

One strategy to alleviate this issue is to reduce the number of training samples and to introduce sparsity in the samples [3, 24]. Ideally, one maintains only a relatively small set of representative prototypes which capture the discriminative essence of the dataset. This was also theoretically confirmed by Crammer *et al.* [3], who showed that prototype-based methods can be more accurate than nearest neighbor classification. Nevertheless, these methods require rather elaborate learning.

Another successful approach is to focus on sparsity in the variables and perform an efficient low dimensional embedding. For instance, one can accelerate nearest neighbor search by performing a binary Hamming embedding. This can be done by applying hashing functions directly [10] or on kernelized data [13]. In particular, hyperplanes or hyperspheres are used to partition the data. Data independent variants such as [6, 2] ignore the structure of the data at all. Data dependent methods [22, 8] consider the structure of the data, however these mostly build on an isotropic cluster assumption and thus do not exploit the general structure of the data.

In contrast, similar to [10] we want to exploit a general metric structure for hashing. Thus, the goal of this paper is to bridge efficient training and efficient evaluation in context of Mahalanobis metric learning. In particular, we build on an efficient metric learning approach, namely KISSME [11], which has shown competitive results on a range of benchmarks, and adopt it for two different hashing strategies. In addition, we introduce a metric-based re-ranking strategy, which further improves the classification results. The proposed approach finally enables us to drastically reduce the computational effort during training and evaluation while maintaining accuracy.

The rest of this paper is structured as follows. In Sec. 2 we first summarize the main ideas of the used metric learning approach and hashing in general and then show how both approaches can be integrated for efficient (image) retrieval. Succeeding, in Sec. 3 we show detailed experimental results on standard machine learning datasets and on the challenging PubFig [15] face recognition benchmark

and also give a comparison to state-of-the-art hashing methods. Finally, in Sec. 4 we summarize and conclude the paper.

## 2 KISS HASH

In the following, we introduce our new metric-based k-NN classification scheme taking advantage of both, efficient learning and evaluation. The main idea is to efficiently learn a Mahalanobis metric, which better captures the intrinsic structure of the feature space, and to approximate it using hashing techniques.

The main goal of hashing is to reduce the classification effort by using a more compact representation. In particular, by mapping the features from a $d$-dimensional original space to a lower $m$-dimensional space, where $m \ll d$. A widely used approach is to apply a Hamming embedding, where the data is represented in form of binary strings. This allows for comparing the data via XOR operations, which can be computed efficiently by special purpose instructions on modern computer hardware. Given a sample $\mathbf{x}$, its binary hash-code $\mathbf{h}$ $(m \times 1)$ can be obtained via

$$\mathbf{h}(\mathbf{x}) = \operatorname{sign}\left(\mathbf{P}\mathbf{x} + \mathbf{t}\right) \ , \tag{1}$$

where $\mathbf{P}$ is a hashing matrix $(m \times d)$ and $\mathbf{t}$ $(m \times 1)$ is a threshold vector.

As minimization of the distances in Hamming space is related to the minimization of the distances in original space, in the following we derive two embedding strategies, exploiting the information captured by a Mahalanobis distance. The only requirement for this relation is that the hashing function sustains the locality sensitive hashing (LSH) requirement [6, 2] that the probability of a collision in the hash table is related to the similarity in the original space. In the following, we first describe how to efficiently obtain a Mahalanobis metric in Sec. 2.1 and then derive two different metric-based hashing strategies: (a) via random hyperplane hashing (Sec. 2.2) and (b) via eigen-hashing (Sec. 2.3). In addition, in Sec. 2.4 we introduce a re-ranking scheme for hashing.

### 2.1 Efficient Mahalanobis Metric Learning

In general, the goal of Mahalanobis distance learning is to learn a distance function $d^2_{\mathbf{M}}$, which measures the squared distance between two data points $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$:

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j) \ , \tag{2}$$

where $\mathbf{M}$ induces a valid pseudo metric if it is a symmetric positive semi-definite matrix. Several different approaches (e.g., [20], [4], or [7]) have been proposed and are widely applied for various tasks. However, such approaches require complex iterative, computationally expensive optimization schemes, making them often infeasible for large-scale problems. To overcome these limitations, *KISS metric* learning (KISSME) [11] builds on a statistical motivated formulation that allows for learning just from equivalence constraints.

For the following discussion let $\mathbf{x}_i$,$\mathbf{x}_j \in \mathbb{R}^d$ be a pair of samples and $y_i, y_j \in \{1, 2, \ldots, c\}$ the labels. Further we define a set of similar pairs $\mathcal{S} = \{(i,j)\,|y_i = y_j\}$ and a set of dissimilar pairs $\mathcal{D} = \{(i,j)\,|y_i \neq y_j\}$. The goal of KISSME is to decide whether a pair $(i,j)$ is similar or not. From a statistical inference point of view the optimal statistical decision can be obtained by a likelihood ratio test. Hereby, the hypothesis $H_0$ that the pair is dissimilar is tested against hypothesis $H_1$ that the pair is similar:

$$\delta(\mathbf{x}_{ij}) = \log\left(\frac{p(\mathbf{x}_{ij}|H_0)}{p(\mathbf{x}_{ij}|H_1)}\right) = \log\left(\frac{f(\mathbf{x}_{ij}|\theta_0)}{f(\mathbf{x}_{ij}|\theta_1)}\right) \ , \tag{3}$$

where $\delta$ is the log-likelihood ratio, $f(\mathbf{x}_{ij}|\theta)$ is a pdf with parameters $\theta$ and $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.

Thus, KISSME casts the metric learning problem into the space of pairwise differences, as also the similarity Eq. (2) is defined via pairwise differences. This space has zero-mean and is invariant to the actual locality of the samples in the feature space. Assuming zero-mean Gaussian distributions within the difference space Eq. (3) can be re-written to

$$\delta(\mathbf{x}_{ij}) = \log\left(\frac{\frac{1}{\sqrt{2\pi|\Sigma_\mathcal{D}|}}\exp(-1/2\ \mathbf{x}_{ij}^T\ \Sigma_\mathcal{D}^{-1}\ \mathbf{x}_{ij})}{\frac{1}{\sqrt{2\pi|\Sigma_\mathcal{S}|}}\exp(-1/2\ \mathbf{x}_{ij}^T\ \Sigma_\mathcal{S}^{-1}\ \mathbf{x}_{ij})}\right), \tag{4}$$

where $\Sigma_\mathcal{S}$ and $\Sigma_\mathcal{D}$ are the covariance matrices of $\mathcal{S}$ and $\mathcal{D}$, respectively.

The maximum likelihood estimate of the Gaussian is equivalent to minimize the distances from the mean in a least squares manner. This allows KISSME to find respective relevant directions for $\mathcal{S}$ and $\mathcal{D}$. By taking the log and discarding the constant terms we can simplify Eq. (4) to

$$\delta(\mathbf{x}_{ij}) = \mathbf{x}_{ij}^T\ \Sigma_\mathcal{S}^{-1}\ \mathbf{x}_{ij} - \mathbf{x}_{ij}^T\ \Sigma_\mathcal{D}^{-1}\ \mathbf{x}_{ij} = \mathbf{x}_{ij}^T(\Sigma_\mathcal{S}^{-1} - \Sigma_\mathcal{D}^{-1})\mathbf{x}_{ij}\ . \tag{5}$$

Finally, the Mahalanobis distance matrix $\mathbf{M}$ is obtained by

$$\mathbf{M} = \left(\Sigma_\mathcal{S}^{-1} - \Sigma_\mathcal{D}^{-1}\right)\ . \tag{6}$$

### 2.2 Hashing by random hyperplanes

As the metric matrix $\mathbf{M}$ is positive semi-definite (p.s.d.) we can decompose it as $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$ by Cholesky factorization. The matrix $\mathbf{L}$ can be seen as linear transformation that scales and rotates the feature space according to $\mathbf{M}$. After applying the linear transformation one can perform standard locality sensitive hashing techniques as random hyperplane hashing.

Thus, to obtain the hash value for a single bit $h_i$ the feature vector $\mathbf{x}$ is first transformed by $\mathbf{L}$ and then projected onto a random vector $\mathbf{r}_i$ that is drawn from a Gaussian distribution with zero mean and unit variance:

$$h_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{r}_i^\top \mathbf{L}\mathbf{x} \geq t_i \\ -1 & \text{otherwise .} \end{cases} \tag{7}$$

Let

$$\mathbf{R}_m = [\mathbf{r}_1 \dots \mathbf{r}_m] \tag{8}$$

be a matrix composed of $m$ random vectors, where $m$ is the desired dimensionality. Then, according to Eq. (1) we can re-formulate Eq. (7) such that a hash code $\mathbf{h}(\mathbf{x})$ over all feature dimensions can be estimated:

$$\mathbf{h}(\mathbf{x}) = \operatorname{sign}\left(\mathbf{R}_m^\top \mathbf{L}\mathbf{x} + \mathbf{t}\right) . \tag{9}$$

### 2.3 Hashing by eigen-decomposition

Since $\mathbf{M}$ is p.s.d. we can also perform an eigen-decomposition $\mathbf{M} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$. This allows us to hash with eigenvectors $\mathbf{v}_i$ as follows:

$$h_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{v}_i^\top \mathbf{x}_i \geq t_i \\ -1 & \text{otherwise .} \end{cases} \tag{10}$$

Again, let

$$\mathbf{V}_m = [\mathbf{v}_1 \dots \mathbf{v}_m] \tag{11}$$

be the matrix containing the eigenvectors associated with the largest eigenvalues, we can estimate an $m$-dimensional hash code for the the feature vector $\mathbf{x}$ by

$$\mathbf{h}(\mathbf{x}) = \operatorname{sign}\left(\mathbf{V}_m^\top \mathbf{x} + \mathbf{t}\right) . \tag{12}$$

### 2.4 Retrieval of hashed Examples

The Hamming embedding enables a very efficient search based on the compact binary representation. Further, on modern CPUs special purpose instructions exist that are even able to calculate the Hamming distance in a few clock-cycles. Also approximate search strategies exist that are tailored to the search in Hamming space (*e.g.*, [2] or [16]).

For the proposed method the focus is on short binary codes that can be efficiently matched followed by a re-ranking step. In particular, a short list of samples is generated by searching in Hamming space, which is then used for exact k-NN with the learned metric. To ensure efficiency compact codes are used in the first step and only a rather small subset of samples is re-ranked. In particular, we aim at re-ranking $\mathcal{O}(N^{\frac{1}{1+\epsilon}})$ samples, where $N$ is the number of training samples in the respective dataset. For instance, if $\epsilon = 1$ only $\mathcal{O}(\sqrt{N})$ samples have to be checked. Thus, for higher values of $\epsilon$ less samples have to be re-ranked.

## 3 Experiments

To show the applicability of our method we conduct experiments on three standard benchmarks and on the Public Figures [15] face recognition benchmark. The goals of our experiments are twofold. First, we want to show that with a drastically reduced evaluation effort we are able to obtain similar results to KISSME and other metric learning baselines. Second, we want to prove that we are competitive to state-of-the-art hashing schemes, requiring less effort.

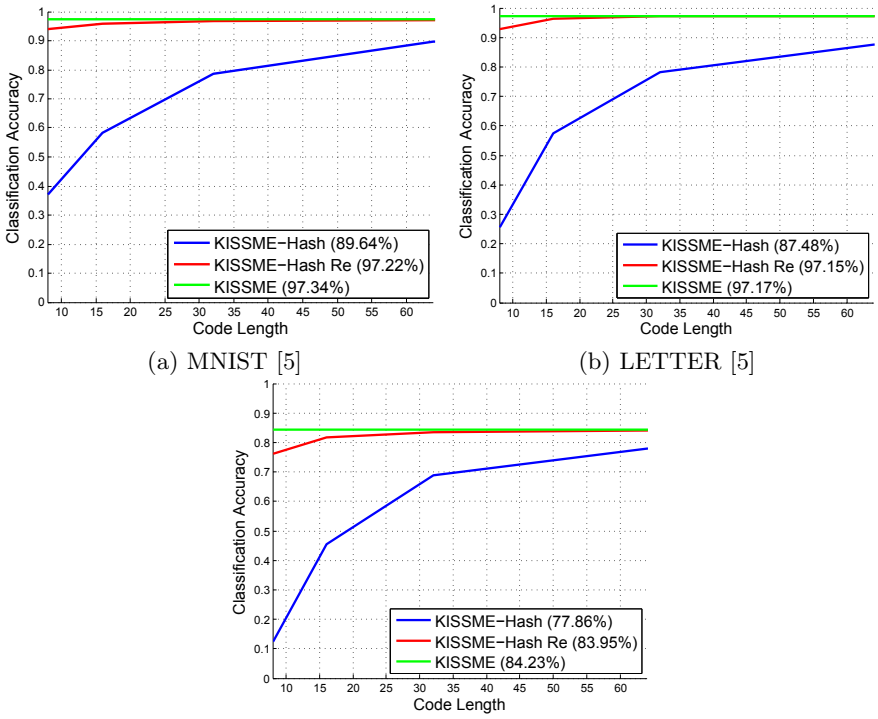
### 3.1 Machine Learning Databases

In the following, we benchmark our proposed method on MNIST [5], LETTER [5] and CHARS74k [1]. First, we give a brief overview of the databases. Second, we compare the performance related to the evaluation complexity between our method and other hashing approaches.

The MNIST database [5] of hand written digits contains in total 70,000 images in one train-test split. 60,000 samples are used for training and 10,000 for testing. The images have a resolution of $28 \times 28$ pixels and are in grayscale. In contrast, the LETTER [5] database contains a large number of synthesized images showing one of the 26 capital letters of the English alphabet. The images are represented as 16 dimensional feature vector which describes statistical moments and edge counts. Chars74K [1] contains a large mixed set of natural and synthesized characters. The images comprise one of the 26 capital or lowercase letters and digits, respectively. 7,705 characters are cropped of natural images, 3,410 are hand drawn and 62,992 are synthesized. Further, the database is split into one train/test set where 7400 samples are organized for testing and the rest for training.

In Figure 1 we compare our random hyperplane hashing method to its baseline on MNIST, LETTER, and CHARS74k. Therefore, we plot the 1-NN classification error in relation to the code length, where the maximum code length is restricted to 64 bits. In particular, we report the following results: (a) Standard KISSME without hashing, (b) nearest neighbor search in Hamming space, and (c) nearest neighbor search in Hamming space with short list re-ranking. For the re-ranking step we fix $\epsilon$ to 1, retrieving $\mathcal{O}(\sqrt{N})$ samples, which is roughly 1% of samples in these cases.

For the following discussion we focus on the results on MNIST of our random hyperplane based hashing method. The results for MNIST are visualized in Figure 1 (a), although the relative results are comparable on the different datasets. The direct nearest neighbor search in Hamming space performs initially significantly worse than the short list re-ranking method. By increasing the number of codes the performance gap gets smaller. However, ultimately for MNIST a performance gap of about 7.58% remains with a code length of 64 bits. This confirms the importance of the re-ranking step. If the short list is kept reasonable sized the computational effort is manageable. Comparing KISS-Hash with re-ranking to KISSME reveals that even with short codes comparable per-




(a) MNIST [5] (b) LETTER [5]

(c) CHARS74k [1]

Fig. 1: **Comparison of 1-NN classification accuracy (%) on (a) MNIST (b) LETTER, and (c) CHARS74k for random hyperplane hashing**. Numbers in parentheses denote the classification accuracy with 64 bits.

formance can be obtained. Starting from 16 bits nearly the same performance is reached at a much lower computational cost.

Next, in Table 1 we benchmark our method to various competing methods. In particular, we provide a closer look on different well-established Mahalanobis metric learning methods and hashing schemes. Comparing KISSME to other metric learning methods, *i.e.*, ITML, LDML, and LMNN, reveals that it is competitive in most cases, though requiring drastically less training time. Further, our random hyperplane hashing method as well as the eigenanalysis hashing have very similar performance to KISSME, though drastically reducing the evaluation time. Next, we compare the classification error between our methods and others and relate to their evaluation complexity. For the kernelized hashing approach of [13] the evaluation scales linearly with the number of kernel samples $S$ times the kernel complexity $K_c$: $\mathcal{O}(SK_c)$. In most cases the kernel complexity is similar to a distance evaluation. KLSH requires many kernel samples to obtain similar results, we tested RBF and learned kernels (ITML). The locality-sensitive hashing approach of [10] scales with $\mathcal{O}(MD)$, where $M$ is the length of the short list

| Methods | MNIST | LETTER | Chars74K |
|---|---|---|---|
| **Nearest Neighbors** | | | |
| Nearest Neighbor (1-NN, 3-NN) | 2.92 - 3.09 | 4.30 - 4.35 | 17.97 - 19.99 |
| LMNN $_{\text{3-NN}}$ [19, 20] | 1.70 | 3.54 | 22.89 |
| ITML $_{\text{1-NN}}$ [4] | 2.17 | 4.75 | 17.00 |
| ITML $_{\text{3-NN}}$ [4] | 2.02 | 4.68 | 18.54 |
| LDML $_{\text{1-NN}}$ [7] | 4.04 | 11.25 | 18.62 |
| LDML $_{\text{3-NN}}$ [7] | 3.59 | 10.35 | 20.32 |
| KISSME $_{\text{1-NN}}$ [11] | 2.66 | 2.83 | 15.77 |
| KISSME $_{\text{3-NN}}$ [11] | 2.36 | 2.73 | 18.64 |
| **Locality-sensitive hashing** | | | |
| KISS-HASH-RH $_{\text{1-NN}}$ (64 bit, $\epsilon = 1$) | 2.78 | 2.85 | 16.05 |
| KISS-HASH-EV $_{\text{1-NN}}$ (64 bit, $\epsilon = 1$) | 2.77 | 3.25 | 15.68 |
| KLSH [13, 14] (10,000 kernel samples) | 6.15 | 7.38 | 88.76 |
| Image Search f. Learn. Metrics [10] ($\epsilon = 0.6$) | 5.51 | 8.55 | - |
| Spectral Hashing [22] | 4.25 | 7.42 | 26.03 |
| Multidimensional Spectral Hashing [21] | 5.27 | 33.67 | - |
| Spherical Hashing [8] (256 bit) | 3.19 | 31.4 | 18.59 |

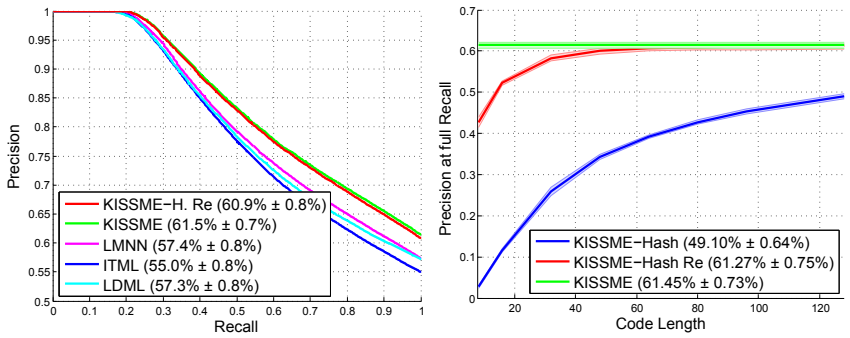
Table 1: **Comparison of classification error rates (%) on MNIST, LETTER and Chars74k**. In particular we provide a closer look on different well-established Mahalanobis metric learning methods and further provide additional results for different locality-sensitive hashing methods.

of samples generated by approximate search in Hamming space [2]. Even at a lower value of $\epsilon$ a performance gap remains. A lower value of $\epsilon$ means to retrieve more samples. Spherical hashing [8] scales with $\mathcal{O}(AD)$ where $A$ is the number of anchor points (code length) where the hyper spheres are anchored. However, it does not match our performance using a comparable number of anchor points.

Recapitulating the different results and relating them to the evaluation complexity of related works reveals that we get competitive results and are more efficient. Moreover, we see that it is beneficial to integrate a metric and to be able to model different scalings and correlations of the feature space.

### 3.2 Public Figures Face Database

In the following, we demonstrate our method for face identification on the Public Figures Face Database (PubFig) [15]. PubFig is a large, real-world face dataset consisting of 58,797 images of 200 people. The evaluation set contains 42,461 images of 140 individuals. PubFig is considered as very challenging as it exhibits huge variations in pose, lighting, facial expression and general imaging and environmental conditions. To represent the faces we use the description of visual face traits [15]. They describe the presence or absence of 73 visual attributes, such as gender, race, hair color. Further, we apply a homogeneous $\chi^2$ feature mapping




(a) (b)

Fig. 2: **Comparison of 1-NN classification accuracy (%) on Public Figures Face Database (PubFig)**. (a) recall / precision by ranking and thresholding classifier scores. Code length of 64 bits, $\epsilon = 1$. (b) Precision at full recall vs code length.

[18]. For the face identification benchmark we organize the data similar to the existing verification protocol in 10 folds for cross-validation. Therefore, we split the images of each individual into 10 disjoint sets.

In Figure 2 (a) we benchmark our random hyperplane hashing to recent Mahalanobis metric learning methods. The results for the eigenvalue hashing are similar. We report the face identification performance in a refusal to predict style. In that sense, recall means the percentage of samples which have a higher classifier score than the current threshold. Precision means the ratio of correctly labeled samples. We use a code length of 64 bits and $\epsilon = 1$. In Figure 2 (b) we report the precision at full recall compared to the code length. In particular, we show that our method generalizes better than LMNN [19], ITML [4] or LDML [7], which require more computational effort in evaluation. At full recall the performance difference to LMNN is 2.50%.

## 4 Conclusion

Mahalanobis metric learning methods have been recently successfully applied for a range of classification problems. However, such approaches have two main drawbacks: High computational effort during (a) training and (b) evaluation. In this paper, we proposed a metric-based hashing method that overcomes both problems. On the one hand side building on an efficient metric learning approach, we obtain competitive classification results on various challenging large-scale benchmarks. On the other hand side, exploiting the learned metric structure by hashing, finally leads to a drastically reduced effort at test time while maintaining the discriminative essence of the data.

**Acknowledgments.** The work was supported by the FFG projects Human Factors Technologies and Services (2371236) and Mobile Traffic Checker (8258408).

## References


1. Campo, T.E.d., Babu, B.R., Varma, M.: Character Recognition in Natural Images. In: Proc. VISAPP (2009)
2. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: ACM symposium on Theory of computing (2002)
3. Crammer, K., Gilad-bachrach, R., Navot, A., Tishby, N.: Margin analysis of the LVQ algorithm. In: Advances NIPS (2002)
4. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Proc. ICML (2007)
5. Frank, A., Asuncion, A.: UCI machine learning repository (2010), university of California, Irvine, School of Information and Computer Sciences. Available online at http://archive.ics.uci.edu/ml
6. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proc. Very Large Data Bases (1999)
7. Guillaumin, M., Verbeek, J., Schmid, C.: Is that you? Metric learning approaches for face identification. In: Proc. ICCV (2009)
8. Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E.: Spherical hashing. In: Proc. CVPR (2012)
9. Hirzer, M., Roth, P.M., Köstinger, M., Bischof, H.: Relaxed pairwise learned metric for person re-identification. In: Proc. ECCV (2012)
10. Jain, P., Kulis, B., Grauman, K.: Fast image search for learned metrics. In: Proc. CVPR (2008)
11. Köstinger, M., Hirzer, M., Wohlhart, P., Roth, P.M., Bischof, H.: Large scale metric learning from equivalence constraints. In: Proc. CVPR (2012)
12. Köstinger, M., Roth, P.M., Bischof, H.: Synergy-based learning of facial identity. In: Proc. DAGM (2012)
13. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing. Trans. PAMI (2012)
14. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: Proc. ICCV (2009)
15. Kumar, N., Berg, A.C., Belhumeur, P.N., Nayar, S.K.: Attribute and Simile Classifiers for Face Verification. In: Proc. ICCV (2009)
16. Norouzi, M., Punjani, A., Fleet, D.J.: Fast search in hamming space with multi-index hashing. In: Proc. CVPR (2012)
17. Shen, C.: Non-sparse linear representations for visual tracking with online reservoir metric learning. In: Proc. CVPR (2012)
18. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. Trans. PAMI 34(3) (2011)
19. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances NIPS (2006)
20. Weinberger, K.Q., Saul, L.K.: Fast solvers and efficient implementations for distance metric learning. In: Proc. ICML (2008)
21. Weiss, Y., Fergus, R., Torralba, A.: Multidimensional spectral hashing. In: Proc. ECCV (2012)
22. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances NIPS (2008)
23. Ye, J., Zhao, Z., Liu, H.: Adaptive distance metric learning for clustering. In: Proc. CVPR (2007)
24. Zhang, Z., Sturgess, P., Sengupta, S., Crook, N., Torr, P.H.S.: Efficient discriminative learning of parametric nearest neighbor classifiers. In: Proc. CVPR (2012)