Metric imitation by manifold transfer for efficient vision applications

Dengxin Dai[1] Till Kroeger[1] Radu Timofte[1] Luc Van Gool[1,2]
[1]Computer Vision Lab, ETH Zurich [2] VISICS, ESAT/PSI, KU Leuven
{dai, kroegert, timofter, vangool}@vision.ee.ethz.ch

Abstract

Metric learning has proved very successful. However, human annotations are necessary. In this paper, we propose an unsupervised method, dubbed Metric Imitation (MI), where metrics over cheap features (target features, TFs) are learned by imitating the standard metrics over more sophisticated, off-the-shelf features (source features, SFs) by transferring view-independent property manifold structures. In particular, MI consists of: 1) quantifying the properties of source metrics as manifold geometry, 2) transferring the manifold from source domain to target domain, and 3) learning a mapping of TFs so that the manifold is approximated as well as possible in the mapped feature domain. MI is useful in at least two scenarios where: 1) TFs are more efficient computationally and in terms of memory than SFs; and 2) SFs contain privileged information, but are not available during testing. For the former, MI is evaluated on image clustering, category-based image retrieval, and instance-based object retrieval, with three SFs and three TFs. For the latter, MI is tested on the task of example-based image super-resolution, where high-resolution patches are taken as SFs and low-resolution patches as TFs. Experiments show that MI is able to provide good metrics while avoiding expensive data labeling efforts and that it achieves state-of-the-art performance for image super-resolution. In addition, manifold transfer is an interesting direction of transfer learning.

1. Introduction

The problem of computing good metrics is of great theoretical and practical interest in information processing. Applications include clustering [52,53], image annotation [24], retrieval [19, 31], classification [2, 19, 51], among others. It is often addressed by metric learning [1, 24, 28, 42, 51], with great success in the past years. The drawback of metric learning is that it requires some form of human annotation (*e.g*. full category labels [2, 19, 37, 51], pairwise constraints [1, 28, 31, 44]), which is expensive to acquire.

To address this problem, this paper aims to learn good metrics for cheap features without using human annotation. We draw inspiration from transfer learning and propose a novel approach, coined Metric Imitation (MI). MI takes state-of-the-art, off-the-shelf features as source features (SFs) and cheap features as target features (TFs) to learn good metrics for the TFs by imitating the metrics computed over the SFs. MI is a general framework and can at least be applied to: 1) efficiently learning good TF metrics when SFs are more powerful than TFs, but are computationally more expensive and/or more memory-hungry; and 2) creating good metrics for TFs when SFs contain privileged information but are not available at testing time.

MI comes out of a marriage of advances in metric learning and transfer learning. On the one hand, metric learning now is able to learn good metrics over general features for specific tasks [30], with human supervision. On the other hand, transfer learning can now transfer knowledge (often classifiers) learned in one domain to another [41], for which no further human supervision is necessary. Observing both developments then begs the question whether metrics can be computed over one feature (*i.e*. SFs) to then be transferred to the domain of another feature (*i.e*. TFs) and automatically supervise the metric learning process there. In this paper, we demonstrate this for several vision tasks. The main advantages of the method are: 1) it performs in an unsupervised manner, i.e. without human annotation; 2) it can be efficient as only TFs are needed during test time; 3) it can inject domain knowledge carried through SFs to TFs for metric computation.

The metric is learned in the framework of Mahalanobis distance learning, *i.e*. a linear mapping function of TFs is learned and applied prior to performing the Euclidean distance metric. Specifically, the method works as follows: 1) it translates the properties of metrics computed over SFs into manifold geometries; 2) it transfers the manifold to the domain of the TFs as view-independent properties; and 3) it learns a mapping function of the TFs so that the transferred manifold is approximated as well as possible in the transformed space. By doing this, the *neighborhood properties* of data computed over the SFs are preserved in the transformed space of TFs, *i.e*. close neighbors are still close. The reason for ensuring this is that neighbors search is crucial in many vision applications, such as clustering, retrieval, and classification.

In accordance with our previous remarks, the usability of MI is validated in two scenarios of metric learning: 1) compute good metrics for cheap features; and 2) transfer privileged information. For the first scenario, MI was tested on instance-based object retrieval using the INRIA Holiday dataset [27] and the UKbench dataset [38], and on category-based image retrieval and image clustering using four other datasets: Scene-15 [34], CUReT-61 [11], Caltech-101 [15], and Event-8 [35]. Three sophisticated, high-dimensional features were used as the SFs: object-bank [36], SIFT-llc [50], and the CNN feature [6], and three general, cheap features used as the TFs: GIST [40], LBP [39], and PHOG [4]. Extensive experiments show that MI consistently and significantly outperforms the metrics computed directly over the same TFs, while getting very close to the metrics from the computationally more expensive SFs in some cases. For the second scenario, MI was evaluated on example-based image super-resolution. Patches of high-resolution images, which are not available at testing time, are used as SFs and patches of low-resolution images as TFs. Experiments show that MI is able to improve the performance of k-NN-based methods for image super-resolution.

2. Related Work

Our method is generally relevant to Metric Learning, Feature Embedding, and Transfer Learning.

2.1. Metric Learning

There has been a great deal of research devoted to metric learning to tune distance functions with human supervision. Typically, metric learning methods follow as guiding principle to learn a mapping that shrinks the distances between similar objects, while expanding the distance between dissimilar objects. Since a comprehensive overview is beyond the scope of this paper, we summarize the most relevant ones according to the types of annotation used. Readers are referred to [30] for a survey.

Class label is one of the most popular annotation types for metric learning. Notable examples include Neighborhood Component Analysis (NCA) [21], Collapsing Classes (CC) [20] and Large-Margin Nearest Neighbors (LMNN) [51]. NCA maximizes the expected number of correctly retrieved points under a stochastic neighbor selection rule. CC optimizes a similar stochastic neighbor selection rule while attempting to collapse each class to a single point. The idea enforces more regularity on the output space than NCA and leads to a convex optimization problem, but the assumption that entire classes can be collapsed to distinct points rarely holds in practice. LMNN [51] relaxes this strong assumption by defining target neighbors as the k closest samples in the same class, and maximizes margins between distances to target neighbors and to other points.

Pairwise Constraints correspond to another popular type of annotation used in metric learning, in which similar (dissimilar) pairs of objects are indicated. Earlier work by Xing *et al*. [53] formulates the clustering problem as a semi-definite programming task under similarity and dissimilarity constraints. With similar annotations, Davis *et al*. [12] tackle the metric learning problem with an information theoretic approach, namely by minimizing the Kullback-Leibler divergence between a learned and a prior matrix, expressing the pairwise constraints. Guillaumin *et al*. [25] offer a probabilistic view on learning distance metrics with pairwise constraints and solve the problem by MAP. Following the spirit of [12,25], Kostinger *et al*. [28] present the KISS framework by formulating the problem as a likelihood ratio test, which avoids complex optimization problems.

While all these methods perform well, human annotations have to be provided. As a result, methods requiring weaker or no supervision are desired. Our method offers this by learning metrics over cheap features by imitating the standard metrics computed over other, sophisticated off-the-shelf features, which either contain privileged information or are more useful for the tasks at hand but too computationally expensive.

2.2. Embedding

There is a great deal of work about embedding for nonlinear dimensionality reduction, with the philosophy that many high-dimensional data can be characterized by a low-dimensional nonlinear manifold. The methods broadly fall into two categories: those providing a mapping function for out-of-sample extension and those just offering a visualization. We will briefly overview the methods which provide mapping functions. Embedding methods proceed mainly in two steps: 1) quantify manifold properties (local geometry); and 2) seek a low-dimensional space so that the learned properties are mostly preserved. A large range of manifold properties have been considered. For instance, Multidimensional Scaling [7] and its extension Isomaps [45] are designed to capture and preserve pairwise distances between points. Locally-linear embedding (LLE) [26, 43] computes the local linearity of the manifold and attempts to preserve that in the low-dimensional space. Laplacian Eigenmaps (LapEigen) [3], and Hessian LLE (HLLE) [14]. proceed in a similar manner as LLE does, but the properties captured are the locality of the manifold, and the local curvature of the manifold. Due to space limitations interested readers are referred to the comparative overview in [48].

Our approach draws inspiration from and extends the traditional embedding methods. The method imitates the manifold properties obtained from SFs by learning a transformation for the TFs, trading off a slight drop in precision against a significant gain in efficiency. The method also avoids feature embedding in the high-dimensional space of SFs, which is time-consuming.

2.3. Transfer Learning and Domain Adaptation

Our method also is akin to transfer learning. Transfer learning addresses the problem of transferring the knowledge learned in a source domain to a different target domain. Successful applications in vision include knowledge transfer from one data source to another (*e.g.* videos to images) [17, 23, 32] and knowledge transfer from known classes to unseen classes [33]. The difference between such work and ours is in the form of the knowledge. These methods transfer definitions of classes (learned classifiers) to a new domain where few or no training data is available, by reducing the discrepancy from source domain to target domain. Our goal, however, is to transfer the manifold structure from our source domain to a target domain to imitate the metrics computed in the source domain, where features in the source domain are kept intact. This distinction leads to different algorithms and applications.

3. Metric Imitation

In this section, we will introduce Metric Imitation (MI) in detail. Like most metric learning or embedding methods, MI operates in two modes: training and mapping. Training learns the projection transformation matrix A guided by the transferred manifold, while 'mapping' performs the learned transformation for better distance metrics during testing.

The generic problem of MI can be summarized as follows. Given a set of n training images $\mathcal{D} = \{I_1, I_2, ..., I_n\}$, and their corresponding SFs $\mathcal{D}_s = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \in \mathbb{R}^{m \times n}$ and TFs $\mathcal{D}_c = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n\} \in \mathbb{R}^{q \times n}$ (often $q \ll m$), the goal is to find a transformation matrix $A \in \mathbb{R}^{q \times q}$ that maps the n TFs $\mathbf{y}$'s to n transformed points $\bar{\mathcal{D}}_c = \{\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2, ..., \bar{\mathbf{y}}_n\}$, where $\bar{\mathbf{y}}_i = A^T \mathbf{y}_i$. The guiding principle of the learning is to approximate the manifold defined by all $\mathbf{x}$'s by the manifold defined over all $\bar{\mathbf{y}}$'s. Note that the transformation matrix A can be a low-rank matrix for the purpose of further dimension reduction. We force it to be a full-rank matrix, as the dimensionality of $\mathbf{y}$ is itself quite low.

3.1. Training

The training part comes with two components: learning the intrinsic manifold over the input $\mathbf{x}$'s, and learning a transformation that has to be applied to the $\mathbf{y}$'s such that the 'intrinsic' manifold is approximated as closely as possible in the learned space of $\bar{\mathbf{y}}$'s.

1. **Learning the intrinsic manifold over SFs**: the aim of this step is to capture the 'intrinsic manifold', i.e. the manifold underlying the input data. Since visual data is highly nonlinear, its global manifold layout tends to be captured better by local properties, such as Local Linear Embedding (LLE) [43], Laplacian Eigenmaps (LapEigen) [3], or Hessian LLE (HLLE) [14]. Compared to methods for global properties, such as Isomap [45], local ones have several advantages. They typically yield better results, and are faster when advantage is taken of matrix sparsity. Local methods share that they first induce a local neighborhood structure over the input data, and then quantify the corresponding local properties over the neighborhood structure. In these methods, the learned manifold is used for dimensionality reduction [3, 14, 43]. However, we use the manifold as a guide to learn a transformation of other features, which are much cheaper to compute and store, so that the learned manifold is approximated with good precision in the transformed space. Below come the two steps of the learning.

(a) Constructing an adjacency graph: by taking each image I in $\mathcal{D}$ as a node, the adjacency graph $G = (\mathcal{D}, \mathcal{E})$ can be constructed in the following way: a directed edge from node i to node j is added, *i.e.* $e_{ij} = 1$, if I_j is among the K nearest neighbors of I_i. The features in source domain $\mathbf{x}$ are used to compute the neighbors under the L_2 distance, but any distance measure can be applied. The global structure of the nonlinear manifold is captured by the interaction of these overlapping local structures. The largest connected component is used for further analysis if the graph is not fully connected.

(b) Quantifying the local properties of the manifold: The local properties (relations) of the manifold can be quantified in a variety of ways, including local linearity as in LLE [26,43], local pairwise distance as in LapEigen [3], local 'curviness' as in HLLE [14], etc. Here, we use two of the most representative ones: local linearity and local pairwise distance. For the reason of efficiency, their linear variants are used.

- LLE [43] assumes that the manifold is locally linear, *i.e.* a data point $\mathbf{x}_i$ is characterized by encoding $\mathbf{x}_i$ as a linear combination W_i (reconstruction weights) of its k nearest neighbors $\mathbf{x}_{ij}$. The weights W are learned by minimizing the following reconstruction error:

$$\begin{aligned} \underset{W}{\text{minimize}} \sum_{i=1}^{n} \|\mathbf{x}_i - \sum_{j=1}^{n} W_{ij}\mathbf{x}_j\|^2 \\ \text{s.t.} \sum_{j=1}^{n} W_{ij} = 1 \end{aligned} \tag{1}$$

where W_{ij} is only allowed to have non-zero value if $e_{ij} = 1$.

- LapEigen [3] characterizes the local manifold structure by encoding the pairwise distances between close neighbors. The neighbors are defined also over the graph G. The weight matrix W is defined as a Gaussian kernel function over the distance of neighboring nodes to reflect the proximity of data points:

$$w_{ij} = \begin{cases} \exp - \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}, & \text{if } e_{ij} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where σ modulates the decreasing rate of W with the distance.

Metric Imitation (MI) learns the structure of the manifold underlying the $\mathbf{x}$'s and describes its local properties through the weight matrix W. The learned W is then transferred to the domain of the $\mathbf{y}$'s. There a linear mapping is learned, i.e. a matrix $A \in \mathbb{R}^{q \times q}$ to be applied to the $\mathbf{y}$'s in order to mimic corresponding manifold characteristics: 1) for LLE, a transformation is learned for $\mathbf{y}$'s so that all mapped data points $\bar{\mathbf{y}}$'s are reconstructed well from their neighbors (defined in graph G) with the learned reconstruction weight W; 2) for LapEigen, MI learns a projection so that all nearby points in the space of the $\mathbf{x}$'s are still nearby points in the mapped space of the $\bar{\mathbf{y}}$'s.

2. **Approximating the manifold by using the TFs**: with the learned manifold property W from SFs (in the manner of LLE or LapEigen), the aim of this step is to compute a linear projection matrix A for TFs so that the manifold properties learned over the SFs are preserved as much as possible in the transformed space. Specifically, MI minimizes the following cost function:

$$\underset{\bar{\mathbf{y}}}{\text{minimize}} \sum_{i=1}^{n} \|\bar{\mathbf{y}}_i - \sum_{j=1}^{n} W_{ij}\bar{\mathbf{y}}_j\|^2. \quad (3)$$

The cost function is very similar to that in Eq.1, but here the weight matrix W is fixed and the optimization is performed over the $\bar{\mathbf{y}}$. For the sake of efficiency, $\bar{\mathbf{y}}$ is constrained to be a transformed version of $\mathbf{y}$ by a linear transformation: $\bar{\mathbf{y}} = A^T\mathbf{y}$. According to the induction of [26], solving Eq.3 can be reduced to solving the following generalized eigenvector problem:

$$YMY^T\mathbf{a} = \lambda YY^T\mathbf{a}, \quad (4)$$

where

$$\begin{aligned} Y &= (\mathbf{y}_1, ..., \mathbf{y}_n), \\ M &= (I - W)^T(I - W), \\ I &= diag(1, ..., 1). \end{aligned}$$

Solving Eq.4 leads to q eigenvectors $\mathbf{a}_1, ..., \mathbf{a}_q$, with eigenvalues ordered from the smallest to the largest. Thus, the embedding is performed as:

$$\mathbf{y} \to \bar{\mathbf{y}} \doteq A^T\mathbf{y}, \quad (5)$$

where the projection matrix $A = (\mathbf{a}_1, \mathbf{a}_1, ..., \mathbf{a}_q)$.

3.2. Mapping

During the test phase, MI only computes the TFs $\mathbf{y}$'s, for the test images, and then computes the distance metric over them with the learned projection matrix A:

$$d(\mathbf{y}_i, \mathbf{y}_j) = \|\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_j\|^2 = \|A^T\mathbf{y}_i - A^T\mathbf{y}_j\|^2, \quad (6)$$

which is then used for the task at hand. Computing the Euclidean distance in this transformed space equals to computing the following Mahalanobis distance function in the original feature space:

$$\begin{aligned} d(\mathbf{y}_i, \mathbf{y}_j) =& (\mathbf{y}_i - \mathbf{y}_j)^T H(\mathbf{y}_i - \mathbf{y}_j) \\ =& (\mathbf{y}_i - \mathbf{y}_j)^T A^T A(\mathbf{y}_i - \mathbf{y}_j). \end{aligned} \quad (7)$$

The form of Eq.6 is preferred to that of Eq.7, because many approximate, fast nearest neighbor techniques have been developed for the euclidean metric. Since the space is learned to preserve the neighborhood properties of the 'intrinsic' manifold, superior performance over common metrics on original TFs is expected. Although our mapping and the embedding for dimensionality reduction are similar technically, they serve totally different purposes. The advantages of MI over the embedding for dimensionality reduction of SFs are twofold: 1) computationally more efficient, as SFs are not needed by MI during testing, and 2) more robust, as fewer parameters are needed to learn the projection, as $q \ll m$. The advantage of MI over the embedding for TFs is that the 'privileged' information in the source domain is injected into the target domain at little extra cost.

4. Experiments

We evaluated Metric Imitation (MI) in the two aforementioned scenarios: 1) learning of good metrics when the source features (SFs) are more powerful but computationally more expensive than the target features (TFs); 2) when SFs contain privileged information but are not available during test time. For the first, MI was evaluated on image clustering, category-based image retrieval, and instance-based object retrieval. For the second, MI is evaluated on example-based super-resolution. For all experiments of these four tasks, MI uses identical parameter settings: the size of the neighborhood K was set to 40 and σ in Eq.2 was set to the mean of the L_2 distance between all feature vectors $\|\mathbf{x}_i - \mathbf{x}_j\|^2$, where $i, j \in \{1, ..., n\}$. These values have been set empirically, but our experiments show that MI is not very sensitive to these choices. If needed, a cross-validation on hold-out sets can be considered.

4.1. Image Clustering

As the amount of image data is on a rampant increase, unsupervised methods to group objects into semantically relevant clusters are in great demand. Such clustering is one of the core problems in modern computer vision. In this section, we apply MI to image clustering.

Datasets: The following four popular datasets are used: Scene-15 [34], CUReT-61 [11], Caltech-101 [15], and Event-8 [35]. They contain 4585 images of 15 classes of general scenes, 5612 images of 61 texture classes, 8677 images of 101 object classes, and 1574 images of 8 classes of sports events, respectively.

Features: In order to sufficiently sample the space of possible features, three different SFs and three different TFs are chosen. The SFs are: SIFT-llc [50], object-bank(OB) [36], and the CNN feature [6]. The chosen TFs are: GIST [40], LBP [39], and PHOG [4]. They were chosen because they are popular in the community, come with available code, and are based on different underlying techniques. Another more important reason is that the three SFs are representatives of the state-of-the-art features but computationally expensive (relatively), and the three TFs are low-dimensional and computationally very cheap.

SIFT-llc is a representative image representation based on local descriptors, which utilizes the locality constraints to encode local descriptors and integrates the results by max-pooling. The CNN feature [6] is obtained by vectorizing the convolutional results of the deep convolutional neural network [29], which is trained on the ImageNet dataset. OB performs object detectors as filters and collects the response of these semantic filters over an image. The features were computed as follows. SIFT-llc[1] and OB[2] were computed with their default parameters in the authors' implementations. For the CNN feature, the MatConvNet package [49] was used with a pre-trained CNN model. The convolutional results at layer 16 were stacked as the CNN feature vector, with dimensionality 4096. GIST was computed with only one cell. For LBP, the uniform one was used with PCA for further dimensionality reduction. For PHOG, the 2-layer pyramid was used. The dimensionalities of the six features are listed in Table 1, from which it is evident that the TFs are much lighter and efficient to compute and handle, in line with the aims of MI.

Evaluation: We follow [8, 10, 47] and used Purity (bigger=better) for evaluation. It measures how pure the discovered clusters are according to the ground truth class labels. Spectral Clustering was used on top of MI for the clustering. We compared to Spectral Clustering with the SFs and TFs as inputs. The K-means algorithm was also tried, but it generally produces similar or poorer results than Spectral Clustering. As suggested in [8], we use the χ^2 distance for LBP and PHOG, and the Euclidean distance for GIST. Euclidean distance is used for all the SFs.

Table 1. The dimensionality of the features.

TFs			SFs		
GIST	PHOG	LBP	CNN	SIFT-llc	OB
20	40	59	4096	21504	44604

Results: 50% of the images (without labels) were used for training and the rest used for testing. Table 2 lists the results of all methods on the four datasets over 5 random training-test splits. Due to space limitations only one type of TFs, LBP, is listed here (same for the following experiments). GIST and PHOG generally yield similar trends than LBP, and their outcomes are reported in the supplementary material. It is expected to find that the SFs perform significantly better than the TFs on image clustering. This allows MI to learn 'useful' knowledge from the SFs. From the table, it is evident that MI is able to learn a better metric for the TFs than the common, generic ones (*e.g.* χ^2 and L_2). For instance, on Scene-15 MI improves the purity of LBP from 0.36 to 0.48 if the CNN feature is used as the SFs and LapEigen is chosen. On Event-8, MI improves the purity of LBP from 0.38 to 0.48 when OB is used as the SFs and LLE is used. Similar trends can be observed across all datasets, all SFs, and the two ways of encoding manifold properties. This shows that MI efficiently provides good metrics for image clustering in general, and that the framework is robust to the specific choices of its components.

Across classes: In order to check the generality of MI, we also tested MI in a wilder scenario, where training and test images are from totally different classes. Half of the classes (floored to the closest integer) were used for training and the rest of classes used for testing, *i.e.* 7 classes used for training for Scene-15, 30 for CUReT-61, 50 for Caltech-101, and 4 for Event-8. All methods run over 5 random training-testing splits and the average results are reported in Table 3. The table shows that even when the training images and test images are from different distributions, MI is still able to improve the performance over the standard metrics of TFs. This suggests that MI is also able to work in a fully unsupervised manner. This property is important, especially for online applications where test data is not available beforehand.

Baselines and competing methods: We compared MI also to two baselines and one competing method. The first baseline is to reduce the dimensionality of SFs via PCA, and then regress the reduced SFs by TFs. The method does not work as well as MI, only producing an improvement of 20% to 40% of that of MI, while being much slower as the vari-

[1] http://www.ifp.illinois.edu/~jyang29/LLC.htm

[2] http://vision.stanford.edu/projects/objectbank

Table 2. Purity of clustering by Metric Imitation (MI), where 50% of the images are used for training and the rest for testing.

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	MI_LLE	MI_Lap	SIFT-llc	MI_LLE	MI_Lap	CNN	MI_LLE	MI_Lap	OB
Scene-15	0.36	0.40	0.46	0.49	0.47	0.48	0.69	0.42	0.48	0.54
CUReT-61	0.33	0.44	0.46	0.39	0.33	0.41	0.60	0.31	0.37	0.44
Caltech-101	0.32	0.34	0.34	0.51	0.37	0.36	0.68	0.37	0.35	0.52
Event-8	0.39	0.46	0.46	0.57	0.47	0.47	0.82	0.48	0.48	0.46

Table 3. Purity of clustering by Metric Imitation (MI) across classes, where half of the classes are used for training and others for testing.

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	MI_LLE	MI_Lap	SIFT-llc	MI_LLE	MI_Lap	CNN	MI_LLE	MI_Lap	OB
Scene-15	0.63	0.67	0.70	0.85	0.65	0.66	0.90	0.61	0.59	0.74
CUReT-61	0.62	0.62	0.64	0.65	0.66	0.69	0.77	0.51	0.58	0.68
Caltech-101	0.57	0.62	0.60	0.73	0.59	0.57	0.77	0.64	0.63	0.70
Event8	0.70	0.72	0.74	0.80	0.70	0.72	0.89	0.75	0.73	0.80

Table 4. The computational time (in seconds) of a full matrix of pairwise distance with different features, where MI(X) denotes that Metric Imitation for feature X.

	GIST	PHOG	LBP	**MI**(GIST)	**MI**(PHOG)	**MI**(LBP)	CNN	SIFT-llc	OB
Scene-15	0.41	0.47	0.53	0.41	0.48	0.53	21.92	104.83	215.56
CURet-61	0.61	0.71	0.78	0.62	0.71	0.84	32.68	161.29	325.97
Caltech-101	1.51	1.73	1.96	1.51	1.75	2.01	78.67	388.01	796.86
Event-8	0.05	0.06	0.08	0.05	0.06	0.08	2.76	13.56	26.69

Table 5. MAP of category-based image retrieval by MI, with 50% images used for training and the rest for testing, and recall set to 0.1.

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	MI_LLE	MI_Lap	SIFT-llc	MI_LLE	MI_Lap	CNN	MI_LLE	MI_Lap	OB
Scene-15	0.50	0.54	0.55	0.60	0.58	0.58	0.72	0.56	0.57	0.65
CUReT-61	0.83	0.90	0.87	0.90	0.90	0.90	0.95	0.86	0.84	0.90
Caltech-101	0.35	0.39	0.38	0.57	0.41	0.41	0.79	0.40	0.40	0.60
Event-8	0.44	0.53	0.52	0.70	0.53	0.55	0.89	0.51	0.50	0.60

Table 6. MAP of category-based image retrieval by MI with the concatenation of LBP, GIST and PHOG (LGP) used as the TFs. 50% images are used for training and the rest for testing. Recall is set to 0.1.

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LGP	MI_LLE	MI_Lap	SIFT-llc	MI_LLE	MI_Lap	CNN	MI_LLE	MI_Lap	OB
Scene-15	0.52	0.60	0.61	0.60	0.64	0.64	0.72	0.62	0.63	0.65
CUReT-61	0.84	0.95	0.93	0.90	0.94	0.96	0.95	0.92	0.90	0.91
Caltech-101	0.42	0.48	0.46	0.57	0.51	0.51	0.79	0.48	0.48	0.59
Event-8	0.52	0.63	0.63	0.70	0.65	0.64	0.88	0.60	0.56	0.58

Table 7. MAP of image retrieval by MI on the Holidays and UKbench datasets, when the recall is set to 1.0.

	TFs	**MI**		SFs	**MI**		SFs	**MI**		SFs
	LBP	MI_LLE	MI_Lap	SIFT-llc	MI_LLE	MI_Lap	CNN	MI_LLE	MI_Lap	OB
Holiday	0.38	0.50	0.48	0.66	0.50	0.49	0.72	0.48	0.46	0.48
Ukbench	0.33	0.39	0.38	0.63	0.44	0.39	0.86	0.36	0.38	0.58

ance matrix of the high-dimensional SFs is needed and the regression itself is also computationally heavy. Also, this baseline is to 'reconstruct' the SFs, which is a harder problem than what MI aims to solve; MI learns (imitates) the 'intrinsic' neighborhood behaviors, which matter most for many vision applications. The second baseline is to sample a collection of constraints from the distance matrix of SFs, and to use them, along with TFs, as the input for standard metric learning (ML) methods. We tried three well-known ML methods: LMNN [51], KISS [28], and ML-Rank [37]. None of them performed well, mainly because these ML methods assume the constraints to be noise-free and perform 'strong' supervised learning with them. The sampled constraints, however, are noisy. MI translates the 'oracle' metric to manifold structures in a more robust way by LLE or LapEigen, which makes it more suitable to exploit the weak supervisory information. We also compared our method to [22], where CCA is used to improve one view of the data by using another view of data which contains privileged information. We found that the results are similar to that of our first baseline as both of them mimic the global behavior of SFs. Also, CCA is slower than MI for our tasks, because the correlation matrix of the two features needs to be learned, which has a large number of values.

Time complexity: The computation time for a full pairwise distance metric for the four datasets is shown in Table 4, where MI is compared to the Euclidean distance metric over SFs and TFs directly. The time was measured on a desktop PC, using a single Core i5 with 2.8 GHz. From the table, it is clear that computing metrics is much faster over TFs than over SFs, as they are of much lower dimensions, as shown in Table 1. It can also be observed from the table that MI needs only marginally more computation time than the original TFs, while improving the quality of metrics for the TFs significantly. This proves the usefulness of MI, especially for scenarios with limited computing power and limited memory budget.

4.2. Category-based image retrieval

Another typical test-bed for metric learning is retrieval. In this section, we apply MI to the task of category-based image retrieval, where the goal is to retrieve images of the same category as that of the query image. The same datasets and features are used as for image clustering in Sec.4.1.

Evaluation: Again, 50% of images were used for training (without using labels) and another 50% for testing. The labels of test images were used for evaluation, where each single image was taken as the query once and the mean average precision (MAP) was computed. The recall was set to 0.1 as precision of the top images matters most.

Results: Table 5 shows all the results of the four datasets. It can be observed that MI serves its purpose well; it improves the precision of image retrieval considerably over the common metrics of the low-dimensional TFs. For instance, on CUReT-61 the precision is improved from 0.83 to 0.90, and on Caltech-101 the precision is improved from 0.35 to 0.41. These results show that MI is very helpful for retrieval, because 1) labeled training data is often not available in retrieval as the datasets are often unorganized; 2) an efficient solution is desirable as retrieval is often performed as an interactive task.

Concatenation of all TFs: We also tested the performance of MI when the concatenation of LBP, GIST, and PHOG features is used as TFs. Table 6 shows the results. As can be seen from Table 6 and Table 5 the concatenation does improve the performance of MI over that of using single TFs. The most interesting observation is that MI gets very close to or is on par with the expensive, high-dimensional SFs. This implies that the cheap, low-dimensional features contain a lot of useful information as long as the right transformation is discovered and ensures that they are 'read' in a more useful way. MI is designed exactly for this purpose.

4.3. Instance-based Object Retrieval

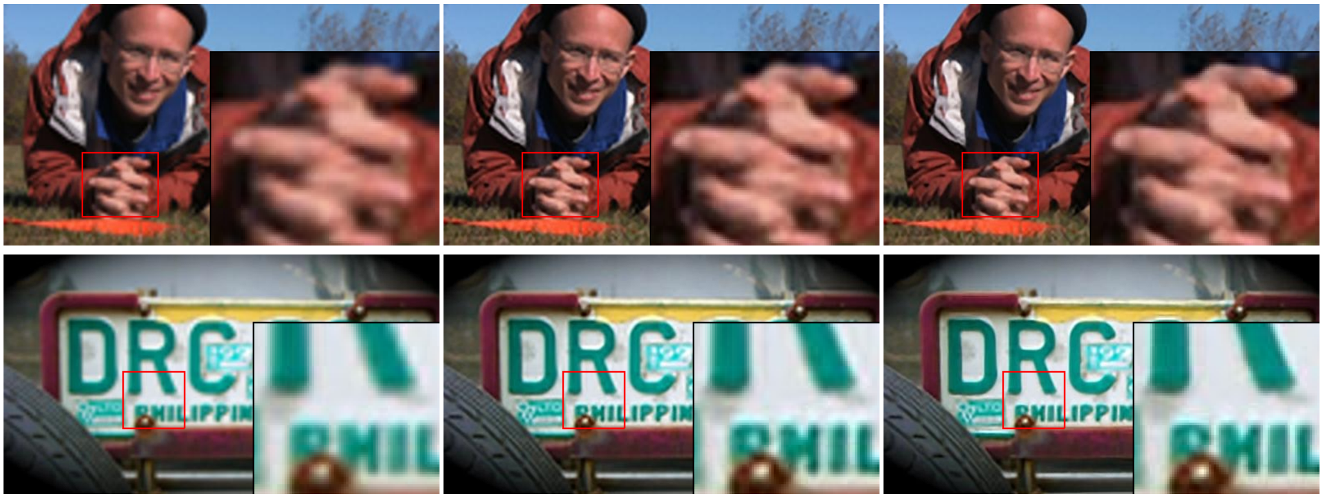
MI was also tested for instance-based object retrieval, where the goal is, given an object instance, to retrieve all instances of the same object class. The latter can be captured from different views, from different distances, etc. We used the same features as for image clustering in Sec.4.1.

Datasets: Two of the most popular datasets for this task were used: the INRIA Holidays dataset [27] and the UKbench dataset [38]. Holidays provides 500 image groups with $1,491$ images in total and for a very large variety of scene types. Each group consists of photos of the same scene/object taken under different conditions: rotations, viewpoint and illumination changes. UKbench contains $10,200$ images of $2,550$ objects, 4 images for each object from different views and distances.

Results: Half of the object classes (floored to closest integer) were used for training (to learn the mapping function) and the rest for testing. MAP was again used as the criterion. The results for LBP are listed in Table 7. It is interesting to see that even when trained on different datasets without using any labels, MI is able to learn good metrics for object retrieval. The learned metrics are significantly better than the standard ones over the TFs. For instance, the MAP of LBP is improved from 0.38 to 0.50 on Holidays and from 0.33 to 0.44 on the UKbench dataset, when the CNN feature is used as the SFs. Yet, there is still room for improvement when compared against the performance of the state-of-the-art object retrieval systems (*e.g.* [42]). There are two main reasons: 1) MI is trained in a fully unsupervised manner: trained on different objects and without using any human annotations; 2) MI uses cheap, low-dimensional features for the sake of efficiency (the difference to [42] in terms of feature dimensionality is as large as 3 orders of magnitude). The goal of MI is different, i.e. providing an efficient solution and avoiding any human annotations.

4.4. Image Super-resolution

The goal of image super-resolution is to generate high-resolution (HR) images from low-resolution (LR) ones. Example-based learning methods have proven successful in learning from a collection of patch pairs: LR patches and corresponding HR patches [5, 9, 13, 46]. In this section, we apply MI to example-based image super-resolution. In order to better show the advantage of MI, we follow the approaches which are directly based on k-NN search [5,9,18]. The very recent method JOR [9] was employed for comparison. JOR jointly learns a collection of regressors from LR patches to HR ones, which collectively yield the smallest super-resolving error for all training data, and selects the most appropriate regressor for each test patch via a voting scheme by its k nearest neighbors from the training samples. MI replaces the standard L_2 metric in the k-NN search by the learned metric, and keeps the rest of the system in-



Bicubic JOR **JOR + MI**

Figure 1. Examples with an upscaling factor ×4. Best seen on the screen. Images are obtained from the Internet.

tact. By the replacement, the method (JOR + MI) is now trying (via imitation) to retrieve neighbors defined over HR patches: searching for LR patches whose corresponding HR patches are similar to the desired HR patch of the test LR patch. This exactly tallies with the goal of example-based image super-resolution.

Since the number of training samples for image super-resolution is much larger (often millions) than that for other vision tasks considered in this paper, solving the eigenvector problem of MI directly is intractable. We employed the eigenfunction technique [16] for an approximate solution. We tested MI with 0.5 million training patches. The method was evaluated on the two standard datasets Set5 and Set14, and MI improves the PSNR from 32.30 to 32.53 on Set5, and from 28.90 to 29.10 on Set14 for an upscaling factor of $\times 3$. For a factor $\times 4$, MI improves JOR from 29.94 to 30.15 on Set5 and from 27.13 to 27.25. The method JOR + MI trained with 0.5 million patches is on par with the JOR trained with 5 million patches [9], but is considerably faster both at training and testing. The method also outperforms the very recent method SRCNN [13] using deep convolutional network, in terms of both performance and speed (in terms of speed at testing, SRCNN is on par with JOR trained from 5 million patches [9]). We also trained the method with 5 million patches, but observed no further improvement. A possible reason is that once samples densely cover the space, standard metrics are sufficient to find good neighbors. We leave the investment to learn more effectively from very large datasets as our future work. However, we believe that the case with 0.5 million patches suffices to prove the concept: MI yields an efficient solution to image super-resolution without sacrificing performance. Two image examples are shown in Fig. 1. The examples and the numbers show that MI improves the results of JOR by transferring domain knowledge from the space of HR patches into that of LR patches.

5. Conclusion

The paper proposed the novel Metric Imitation method to learn good metrics for features in one domain (target features, TFs) by imitating the metrics computed over features of another domain (source features, SFs), and this even without supervision. MI translates the neighborhood behavior of SFs into manifold geometry, and transfers it to the domain of TFs as a guide for metric learning. MI then seeks a linear mapping function of the TFs so that the transferred manifold can be approximated well, which leads to an imitation of the metrics computed over the SFs. The method is easy to understand and easy to implement. The usefulness of MI has been corroborated by two scenarios with four popular vision tasks. Extensive experiments have shown that MI is able to learn significantly better metrics over cheap TFs, while only slightly increasing the time complexity during testing. We hope the idea of metric imitation and manifold transfer will also prove useful for metric learning and transfer learning. The code is available at www.vision.ee.ethz.ch/~daid/MetricImitation.

Acknowledgments. The work is supported by the ERC Advanced Grant VarCity (#273940).

References

[1] M. S. Baghshah and S. B. Shouraki. Semi-supervised metric learning using pairwise constraints. In *IJCAI*, 2009. 1

[2] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using

class specific linear projection. *TPAMI*, 19(7):711–720, 1997. 1

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001. 2, 3, 4

[4] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *ICCV*, 2007. 2, 5

[5] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. *CVPR*, 01:275–282, 2004. 7

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 2, 5

[7] M. A. A. Cox and T. F. Cox. Multidimensional scaling. In *Handbook of Data Visualization*, pages 315–347. 2008. 2

[8] D. Dai, M. Prasad, C. Leistner, and L. V. Gool. Ensemble partitioning for unsupervised image categorization. In *ECCV*, 2012. 5

[9] D. Dai, R. Timofte, and L. Van Gool. Jointly optimized regressors for image super-resolution. In *Eurographics*, 2015. 7, 8

[10] D. Dai, T. Wu, and S. C. Zhu. Discovering scene categories by information projection and cluster sampling. In *CVPR*, 2010. 5

[11] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Trans. Graph.*, 18(1):1–34, 1999. 2, 5

[12] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007. 2

[13] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 7, 8

[14] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc Nat Acad Sci USA*, 100(10):5591–5596, 2003. 2, 3

[15] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision*, 2004. 2, 5

[16] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009. 8

[17] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, 2013. 3

[18] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002. 7

[19] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007. 1

[20] A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *NIPS*, 2005. 2

[21] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *NIPS*, 2004. 2

[22] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011. 6

[23] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, 2011. 3

[24] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009. 1

[25] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *ICCV*, 2009. 2

[26] X. He, D. Cai, S. Yan, and H.-J. Zhang. Neighborhood preserving embedding. In *ICCV*, 2005. 2, 3, 4

[27] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008. 2, 7

[28] M. Kostinger, M. Hirzer, P. Wohlhart, P. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *CVPR*, 2012. 1, 2, 6

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012. 5

[30] B. Kulis. Metric learning: A survey. *Foundations & Trends in Machine Learning*, 5(4):287–364, 2012. 1, 2

[31] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *PAMI*, 31(12):2143–2157, 2009. 1

[32] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, 2011. 3

[33] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 3

[34] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2, 5

[35] L.-J. Li and L. Fei-Fei. What, where and who? classifying event by scene and object recognition. 2007. 2, 5

[36] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 2, 5

[37] B. McFee and G. R. Lanckriet. Metric learning to rank. In *ICML*, 2010. 1, 6

[38] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006. 2, 7

[39] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, 24(7):971–987, 2002. 2, 5

[40] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001. 2, 5

[41] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359, 2010. 1

[42] D. Qin, Y. Chen, M. Guillaumin, and L. Van Gool. Learning to rank bag-of-word histograms for large-scale object retrieval. In *BMVC*, 2014. 1, 7

[43] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 2, 3

[44] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *TPAMI*, 36(8):1573–1585, 2014. 1

[45] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 2, 3

[46] R. Timofte, V. De Smet, and L. Van Gool. Anchored neighborhood regression for fast example-based super resolution. In *ICCV*, 2013. 7

[47] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine. Unsupervised object discovery: A comparison. *IJCV*, 2009. 5

[48] L. J. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. *Tech. Rep., Tilburg Uni.*, 2009. 2

[49] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. 5

[50] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2, 5

[51] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, 2009. 1, 2, 6

[52] L. Wu, S. Hoi, R. Jin, J. Zhu, and N. Yu. Learning bregman distance functions for semi-supervised clustering. *IEEE Trans. Knowl. Data Eng.*, 24(3):478–491, 2012. 1

[53] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *NIPS*, 2003. 1, 2