Stochastic Covariance Compression

Matt J. Kusner, Nicholas I. Kolkin, Stephen Tyree, Kilian Q. Weinberger
Washington University in St. Louis
{mkusner,n.kolkin,swtyree,kilian}@wustl.edu

Abstract

Covariance matrices are an effective way to capture global spread across local interest points in images. Often, these image descriptors are more compact, robust and informative than, for example, bags of visual words. However, they are symmetric and positive definite (SPD) and therefore live on a non-Euclidean Riemannian manifold, which gives rise to non-Euclidean metrics. These are slow to compute and can make the use of covariance features prohibitive in many settings, in particular k-nearest neighbors (kNN) classification. In this paper we present Stochastic Covariance Compression, an algorithm that compresses a data set of SPD matrices to a much smaller set with similar kNN characteristics. We show that we can reduce the data sets to 1/6 and in some cases even up to 1/50 of their original size, while approximately matching the test error of full kNN classification. In fact, because the compressed set is learned to perform well on kNN tasks, it sometimes even outperforms the original data set, while requiring only a fraction of the space and drastically reduced test-time computation.

1. Introduction

Symmetric positive definite (SPD) matrices are often used in computer vision to describe structure tensors (Goh & Vidal [2008]), diffusion tensors (Pennec et al. [2006]) or covariance region. Latter are particularly well suited for the task of object detection from many different viewpoints. Covariance matrices of interest points (covariance features) have also been employed for person tracking in surveillance video (Pang et al. [2008]; Porikli et al. [2006]), given their superior ability to cope with low-resolution images (Harandi et al. [2012]). For scene classification, covariance matrices have shown to be competitive with a number of popular features (Wang et al. [2012]; Xiao et al. [2010]), such as GIST (Oliva & Torralba [2001]) and SSIM (Shechtman & Irani [2007]).

One reason for their popularity is that covariance features provide a more natural way to deal with bags of features than, say, a codebook. Recall that a codebook is built by first computing ‘bags’ of image descriptors for a subset of the training set and then clustering these descriptors into a set of ‘codewords’. Given this codebook, we can arrive at a ‘bag-of-words’ feature representation for any new image by associating image descriptors with codewords. While simple to compute, the codebook representation is potentially sensitive to the number of codewords (clusters). Too few codewords means important image variations may be averaged over, while too many codewords may capture extrinsic image qualities (e.g., lighting, rotation, scale, etc.). In contrast, we can simply calculate a SPD covariance from a bag of image descriptors. These covariance features encode second order information, which captures the global directions of spread within the feature space and is invariant to small perturbations, such as changes in lighting, as well as image rotations (Porikli et al. [2006]).

Using SPD matrices for computer vision tasks can be non-trivial. It must be taken into consideration that SPD matrices lie on a convex half-cone—a non-Euclidean Riemannian manifold embedded inside a Euclidean space. For example, measuring distances between SPD matrices with the straight-forward Euclidean metric ignores the underlying manifold structure of the data and tends to systematically under-perform in classification tasks (Vemulapalli [2013]).

Recently there has been work toward k-nearest neighbors (kNN) with specialized geodesic distances (Cherian et al. [2011]), kernel support vector machines (Jayasumana et al. [2013]), and boosting (Tuzel et al. [2008]), on the SPD manifold. Because of the SPD constraint, these methods often require significantly more time to make predictions on test data. This is especially true for kNN, as the computation of the Riemannian distance along the SPD manifold requires computing an eigen-decomposition for a single pairwise distance. Cherian et al. [2011] improved the running time of test classification by approximating the Riemannian distance with a symmetrized log-determinant divergence (although the asymptotic complexity is the same). In addition, inherent to the kNN classifier is the required O(n) test-time complexity for a training set of size n. Therefore,
for large-scale training datasets, computing the \( k \)-NN decision rule in a timely manner is non-trivial. While Bregman Ball Trees (BBTs) \cite{Cayton2008} can be adapted to the aforementioned divergence to reduce this running time, we are still required to store the entire training set. Additionally, the performance of BBTs deteriorates quickly as the dimension of the SPD matrices increases.

In this paper, we develop a novel technique that is complementary to the above methods for speeding up \( k \)-NN covariance classification. Our method, called Stochastic Covariance Compression (SCC), learns a compressed training set of size \( m \), such that \( m \ll n \), which approximately matches the performance of the \( k \)-NN classifier on the original data. This new data set does not contain original training samples; instead it contains new, artificially generated input samples which are explicitly designed for low \( k \)-NN error on the training data. During test-time, we only find the \( k \)-nearest neighbors among these artificial training samples. This drastically reduces the computation time and shrinks storage requirements. To facilitate our data compression optimization problem we borrow the concept of stochastic neighborhoods, used in data visualization \cite{HintonRoweis2002} \cite{VanDerMaatenHinton2008} and metric learning \cite{Goldbergeretal2004}, and utilize recent results from the machine learning community on data compression in Euclidean spaces \cite{Kusneretal2014}.

We make three novel contributions: 1. we derive SCC, a new method for compression of covariance data; 2. we devise an efficient method for solving the SCC optimization using the Cholesky decomposition; 3. we demonstrate on real-world covariance data that SCC often learns a compressed set that is about \( 1/6 \), and in some cases even only 1/50 of the training set while approximately matching the test error of full \( k \)-NN classification. In fact, because we learn the compressed set explicitly to minimize \( k \)-NN error, in some cases the accelerated test procedure leads to lower test error than the original training data set.

This paper is organized as follows. Section 2 provides background on covariance features and stochastic neighborhoods. Section 3 introduces our method, SCC, for compressing sets of covariance matrix training samples for \( k \)-NN classification. Section 4 compares the performance of SCC to related training set compression methods. Section 5 describes relevant work in classification with covariance features and fast methods for \( k \)-NN. Section 6 concludes and discusses key directions for future work.

2. Background

Covariance features. Given a set of \( d \)-dimensional feature vectors \( \mathcal{F} = \{ x_1, \ldots, x_{|\mathcal{F}|} \} \subset \mathbb{R}^d \) computed from some input image, we define the symmetric positive-definite (SPD) covariance matrix for the image:

\[
    X = \frac{1}{|\mathcal{F}| - 1} \sum_{r=1}^{|\mathcal{F}|} (x_r - \mu)(x_r - \mu)^\top,
\]

where \( \mu = \frac{1}{|\mathcal{F}|} \sum_{r=1}^{|\mathcal{F}|} x_r \). Using the covariance matrix as feature descriptors for classification \cite{Tuzeletal2006} has led to many recent successes in computer vision applications. For example, in texture classification, using covariance features can provide large speedups over histogram features if the original number of features \( d \) is small, while also improving accuracy \cite{Tuzeletal2006}. The ability to encode second order relationships between features and bag of word descriptors has led covariance matrices to be used for person tracking \cite{Poriklietal2006}, face detection \cite{Pangetal2008}, and object recognition \cite{Jayasumanaetal2013}.

One popular classification technique for SPD covariances is the \( k \)-nearest-neighbor (\( k \)-NN) rule, first introduced for vectorial data by \cite{CoverHart1967}. The \( k \)-NN decision rule classifies an unlabeled input by the majority label of its \( k \) nearest training instances. For vectorial data ‘nearness’ is often computed via the Euclidean distance or a learned Mahalanobis metric \cite{WeinbergerSaul2009}. However, the Euclidean/Mahalanobis distance between two vectorized covariance matrices is a poor approximation to their true distance along the manifold of SPD matrices. The most natural measure of distance between two matrices from the SPD manifold is the Riemannian metric \cite{Cherianetal2011}.

Definition 1. Let \( S_d^+ \) be the positive definite cone of matrices of rank \( d \). The Riemannian metric between any two matrices \( X, Z \in S_d^+ \) is \( D_R(X, Z) = \| \log (Z^{-1/2}XZ^{-1/2}) \|_F \).

While the Riemannian metric is the true distance between two covariances along the SPD manifold, it requires an eigenvalue decomposition for every input \( Z \). The metric becomes intractable to compute even for moderately-sized covariance matrices \( e.g. \), computing \( Z^{-1/2} \in S_d^+ \) can be done via a singular value decomposition in \( O(d^3) \) time. To alleviate this computational burden, a distance metric with similar theoretical properties has been proposed by \cite{Cherianetal2011}, called the Jensen-Bregman LogDet Divergence (JBLD),

\[
    D(X, Z) = \log \left| \frac{X + Z}{2} \right| - \frac{1}{2} \log |XZ|.
\]  

\cite{Cherianetal2011} demonstrate that for nearest neighbor classification, using JBLD as a distance has nearly identical performance as the full Riemannian metric but is much faster in practice and asymptotically requires \( O(d^{2.37}) \) computation \cite{CoppersmithWinograd1990}. 


Adopting a tractable distance computation is a critical step toward practical nearest neighbor classification with SPD covariance matrices. However, as with nearest neighbor classification in general, the size of the training set may be a limiting factor for large scale applications. As such we introduce a novel technique for compressing the training set to a fraction of the original number of instances, greatly expanding the size of covariance datasets we can classify with kNN.

Preliminaries. We begin by introducing our notation for our covariance compression technique. First let $X \subseteq \mathbb{R}^{d \times d \times n}$ be a 3rd order tensor of our $n$ training covariance matrices, with corresponding labels $y_1, \ldots, y_n$. Let $X_i$ for $i = 1, \ldots, n$ refer to the $i$th $(d \times d)$-covariance matrix.

Our compression technique is influenced by two prior works in machine learning: stochastic neighborhood embedding (Hinton & Roweis, 2002) and neighborhood component analysis (Goldberger et al., 2004).

Stochastic Neighborhood Embedding (SNE). SNE (Hinton & Roweis, 2002) learns a low-dimensional embedding of a dataset for the purposes of visualization. Given a dissimilarity $d_{ij}$ between any two vectorial inputs $x_i, x_j \in \mathbb{R}^d$, they define a ‘stochastic neighborhood’ by the probability that $x_j$ is assigned as the nearest neighbor of $x_i$, as such,

$$p_{ij} = \frac{e^{-d_{ij}}}{\sum_k e^{-d_{ik}}} = \frac{1}{\Omega_i} e^{-d_{ij}}. \quad (2)$$

Notice that if the dissimilarity is the (scaled) Euclidean distance $d_{ij} = \gamma^2 ||x_i - x_j||^2$, for a scale parameter $\gamma_i^2$, we can think of the stochastic neighborhood in (2) as a set of Gaussian distributions placed on each instance $x_i$. The main insight of SNE is that a good low-dimensional embedding should preserve this stochastic neighborhood. To do so, SNE minimizes the KL-divergence between the neighborhood distributions in the original (high-dimensional) space and the embedded (low-dimensional) space. This technique was improved by Van der Maaten & Hinton (2008) by replacing the Gaussian stochastic neighborhood with Student $t$-distributions in the embedded space.

Neighborhood Components Analysis (NCA). NCA (Goldberger et al., 2004) makes use of a stochastic neighborhood to learn a matrix $A$ for the Mahalanobis distance: $d_{ij} = ||A(x_i - x_j)||$. This metric is learned to improve the performance of kNN by maximizing an approximation of the leave-one-out (LOO) training accuracy for the stochastic nearest neighbor rule (i.e., $k = 1$). The probability of the event that input $x_i$ with label $y_i$ is classified correctly by its nearest neighbor can be stated as

$$p_i = \sum_{k: y_k = y_i} p_{ik}, \quad (3)$$

where we define $p_{ii} = 0$. NCA learns $A$ by maximizing (3) over all inputs $x_i$ in the training set.

3. Covariance compression

In this section we detail our technique: Stochastic Covariance Compression (SCC). SCC uses a stochastic neighborhood to compress the training set from $n$ inputs to $m$ ‘compressed’ covariances. After learning, the original training set can be discarded and all future classifications are made just using the compressed inputs. Since $m \ll n$, the complexity of test-time classification is drastically reduced, from $O(md^2.37)$ to $O(md^2.37)$, where $O(d^2.37)$ is the asymptotic complexity of computing a single JBLD divergence (1).

Compressed set. Assume we are given a training set of $n$ covariance matrices $X \subseteq \mathbb{R}^{d \times d \times n}$ with corresponding labels $y_1, \ldots, y_n$. Let $X_i$ for $i = 1, \ldots, n$ refer to the $i$th $(d \times d)$-covariance matrix. Our goal is to learn a compressed set of $m$ covariance matrices $Z \subseteq \mathbb{R}^{d \times d \times m}$ with labels $y_1, \ldots, y_m$. To initialize $Z$ we randomly sample $m$ covariance matrices from $X$ and copy their associated labels for each $y_j$. Given this initialization, to learn a compressed set, we place a stochastic neighborhood distribution over $X$ and $Z$. Specifically, let the probability that $Z_j$ is the nearest neighbor of $X_i$ be given as follows,

$$p_{ij} = \frac{e^{-\gamma^2 D(X_i, Z_j)}}{\sum_{k=1}^m e^{-\gamma^2 D(X_i, Z_k)}} = \frac{1}{\Omega_i} e^{-\gamma^2 D(X_i, Z_j)} \quad (4)$$

where $D(X_i, Z_j)$ is the JBLD divergence in (1) and we denote the partition function by $\Omega_i$.

Objective. As in Neighborhood Components Analysis, let $p_i$, in (3), with $p_{ik}$ as defined in (4), be the probability that training input $X_i$ is predicted correctly by the compressed inputs $Z$. We would like $p_i = 1$ for all $X_i \in X$, implying that the compressed set perfectly predicts the training set. The KL-divergence between this ideal “1-distribution” and $p_i$ is simply $KL(1||p_i) = -\log(p_i)$. Our objective is to minimize the sum of these KL-divergences with respect to our compressed set of covariance matrices $Z$,

$$\min_Z - \sum_{i=1}^n \log(p_i). \quad (5)$$
Gradients of compressed inputs and scale. We show how this objective can be optimized by repeatedly iterating between a gradient descent step and a projection step onto the SDP half-cone. To compute the gradient, first let the objective in (5) be denoted as $L$. We demonstrate how to compute the gradient $\frac{\partial L}{\partial Z_j}$, for a single covariance $Z_j$. To begin, note that the gradient of the divergence $D(X_i, Z_j)$ in (1) can be written as a difference of matrix inverses (Cherian et al., 2011),

$$\frac{\partial D(X_i, Z_j)}{\partial Z_j} = (X_i + Z_j)^{-1} - \frac{1}{2} Z_j^{-1}.$$  

Using this result, the gradient of the partition function $\Omega_i$ in (4) is

$$\frac{\partial \Omega_i}{\partial Z_j} = -\gamma^2 e^{-\gamma^2 D(X_i, Z_j)} \frac{\partial D(X_i, Z_j)}{\partial Z_j}.$$  

Given these intermediate gradients, the gradient of the objective can be computed,

$$\frac{\partial L}{\partial Z_j} = \sum_{i=1}^{n} \frac{p_{ij}}{p_i} (\delta_{ij} - p_i) \gamma \frac{\partial D(X_i, Z_j)}{\partial Z_j}$$  

where $\delta_{ij} = 1$ if $y_i = y_j$ and is 0 otherwise. Further, we can readily learn $\gamma^2$ by computing $\frac{\partial L}{\partial \gamma^2}$ as such,

$$\frac{\partial L}{\partial \gamma^2} = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{p_{ij}}{p_i} (\delta_{ij} - p_i) D(X_i, Z_j).$$

However, in our experiments we achieve better empirical performance by cross-validating over $\gamma^2$ to select the value yielding the compressed set with the lowest training error.

Practical gradient descent. After computing $\frac{\partial L}{\partial Z_j}$ we could simply project $Z_j$ onto the symmetric positive definite Riemannian manifold. However, this requires computation on the order of an eigenvalue decomposition for each compressed input. Instead, we perform a change of variable and represent each matrix $Z_j$ by its unique Cholesky decomposition (as $Z_j$ is SPD): $B_j^\top B_j$, where $B_j$ is an upper triangular matrix. With this change, we can simply compute the gradient of $L$ with respect to $B_j$ and guarantee that the new learned $Z_j = B_j^\top B$ is SPD. The only change to the gradient occurs in the JBLD divergence term:

$$\frac{\partial D(X_i, B_j^\top B)}{\partial B} = B(X_i + B_j^\top B)^{-1} - B X_i (B_j^\top B X_i)^{-1}.$$  

Computational Complexity. For a single compressed input $Z_j = B_j^\top B$ each step of gradient descent requires $O(d^3)$ to compute $\frac{\partial D(X_i, B_j^\top B)}{\partial B}$ and $O(2.37)$ to compute $D(X_i, Z_j)$. It requires $O(d^3 m)$ to compute $p_{ij}$ and an added $O(m)$ for $p_i$. Thus the overall complexity of $\frac{\partial L}{\partial Z_j}$ is $O(d^3 m^2 n)$. If we wish to compute $\frac{\partial L}{\partial \gamma}$, we may store the computation $\frac{p_{ij}}{p_i} (\delta_{ij} - p_i)$ from $\frac{\partial L}{\partial Z_j}$, then the complexity of $\frac{\partial L}{\partial \gamma}$ is only $O(d^3 mn)$. Algorithm 1 demonstrates the simplicity of SCC. We minimize our objective (5) via conjugate gradient descent\footnote{http://tinyurl.com/minimize-m} A Matlab implementation of SCC is available at: http://anonymized.

4. Results

In this section we compare our algorithm (SCC) to other techniques designed to speed up kNN classification. Specifically, we consider methods for reducing the size of the training set. First, we describe the covariance datasets we evaluate all methods on, also shown in Figure 1.

ETH80 The ETH80 dataset has images of 8 object categories, each pictured with a solid blue background. For each category there are 10 exemplar objects and for each exemplar the camera is placed in 41 different positions. The task is to classify these images into their respective categories. Cherian & Sra (2014) segment the images and use per-pixel color and texture descriptors, based on 3-dimensional Laws texture filters (Laws, 1980). This combined with gradient features and a Laplacian of Gaussian filter results in 19 × 19 covariances, that we employ.

ETHZ The benchmark dataset ETHZ is a low-resolution set of images from surveillance cameras. The images vary in size from $78 \times 30$ to $400 \times 200$. The original task is to identify the person in a given image, from 146 different individuals. We filter the dataset to include only the most popular 50 classes resulting in each individual having between 59 and 356 images (5193 images total). In general, covariances have been shown to be high-accuracy features for person tracking tasks (Harandi et al., 2012). For the ETHZ dataset Cherian & Sra (2014) describe $18 \times 18$ covariances from pixel-wise feature descriptors which we use here.

FERET The FERET face recognition dataset has 3737 gray-scale images of the faces of 399 individuals, oriented at various angles. We limit the dataset to the 50 most popular individuals. Cherian et al. (2011) compute initial feature

Algorithm 1 SCC

1. Input: $X; y$; compressed data set size $m$
2. Initialize $Z$ by class-based sampling $m$ inputs from $X$
3. Learn $Z$ with conj. gradient descent
4. return $Z$
deviation across initialization or sampling we report the average and standard error of the entire training set. For results that depend on random initialization, ETHZ, and ETH80 do not have a defined train/test split. We report results over 5 random runs. As datasets Brodatz Texture, ETHZ, and ETH80 do not have a defined train/test split we report results over 5 different train/test splits.

**Baseline.** We compare our method *Stochastic Covariance Compression (SCC)* against a number of methods aimed at reducing the size of the training set, which we adapt for the covariance feature setting:

1. $k$NN using the full training set (red dotted line).
2. $k$NN using a class-based *subsampled* training set, which we use as initialization (pink-dotted line).
3. CNN [Hart, 1968].
4. FCNN [Angiulli, 2005], modified for covariance data.

Both CNN and FCNN set subsets of the training set that have the same LOO training error as the full training set. For FCNN we must make a modification to accommodate covariance matrix features. Specifically, FCNN requires computing the centroid of each class at regular intervals during the selection. A centroid of class $y$ is given by solving the following optimization,

$$X_y = \arg \min_X \sum_{i:y_i = y} D(X, X_i)$$

where $D(X, X_i)$ is the JBLD divergence. Cherian et al. [2011] give an efficient iterative procedure for solving the above optimization,

$$X_k = \frac{1}{|\mathcal{Y}|} \sum_{i:y_i = y} \left( \frac{X_k^{-1} + X_i}{2} \right)^{-1}$$

where $\mathcal{Y} = \{y_i | y_i = y\}$ is the set of all training set labels equal to $y$. Given an appropriate choice of $X_0$, the iterative procedure is guaranteed to converge to the true centroid (Cherian et al., 2011). With this modification we can readily run FCNN on covariance data.

**Classification error.** Figure 2 compares the test error of SCC to baselines for sizes of the compressed set equal to...
2%, 4%, 8%, and 16% of the training set (for all datasets except FERET for which we run compression ratios: 10%, 20%, 30%, and 40%). Although CNN and FCNN only output a single reduced training set (the final point on each curve) we plot the intermediate test errors of each method at the above compression ratios as well. On each dataset SCC is able to reduce the test error to that of kNN run on the full dataset using less than or equal to 20% of the training data. SCC is always superior to intermediate results of CNN and FCNN, as well as the subsampling initialization. Only on ETHZ and RGBD could SCC not match the accuracy up to significance, however the error rates are only marginally higher. On datasets ETH80 and ETHZ the final outputs of CNN and FCNN are roughly equivalent to the SCC curve. However, one notable downside is that these algorithm have no control on the size of these final sets, which for FERET are as large as 75% (CNN) and 74% (FCNN). In contrast SCC allows one to regulate the compressed set size precisely.

Speedup. Table 1 shows the speedup of SCC over kNN classification using the full training set for various compression ratios. Results that match or exceed the accuracy (up to significance) are in bold. The maximum speedup without increase in error is achieved on the SCENE15 data set, at 45.8×. Even at 16% compression, there is still a speedup of 6.2× across all data sets. Note that the observed speedups are precisely those we would expect for an algorithm whose test time performance is linear in the size of the input.

Training time. Table 2 describes the average training times for SCC. For maximum compression to 2% the training time is on the order of minutes. As the size of the compressed set gets larger the time increases to multiple hours. The entire compression can be done completely offline prior to testing. The SCC training could potentially be sped up significantly through parallelization on clusters and GPUs. The contributions of the training points to the gradient are independent and have a high computation to memory load ratio—an ideal setting for GPU parallelization.

5. Related Work

Image classification for SPD matrices is a growing field with recent promising models. There has been effort toward adapting the class of algorithms in R to Riemannian manifolds (Goh & Vidal, 2008). One idea starts by constructing the tangent space to the manifold. Using this tangent space it can be shown that one can perform LogitBoost on SPD covariances (Tuzel et al., 2008). Two downsides to the tangent method are (a) computing the tangent space is computationally demanding and (b) the tangent space approximation degrades significantly as one moves away from the tangent point. As such, Jayasumana et al. (2013) developed kernelized support vector machines and kernel PCA for SPD covariance matrices without tangent spaces, via mapping them to a Euclidean Hilbert space. Tuzel et al. (2006) first introduce kNN for SPD covariance matrices. Cherian et al. (2011) derive a fast symmetric divergence: the JBLD divergence, whose square root has similar properties to the (true) Riemannian distance (Sra, 2012).

Training set reduction has been considered in the context of kNN for Euclidean data. There are three primary methods: (a) training consistent sampling, (b) prototype generation and, (c) prototype positioning (for a survey see Tousaint, 2002). Training consistent sampling iteratively add inputs from the training set to a reduced ‘reference set’ until the reference set is perfectly classified. This is precisely the technique of Condensed Nearest Neighbors (CNN) (Hart, 1968). There have been a number of extensions of CNN (Gates, 1972; Tomek, 1976; Devi & Murty, 2002), notably Fast CNN (FCNN) (Angiulli, 2005) which has training time linear in the size of the training set (instead of cubic complexity). Prototype generation creates new inputs to represent the training set, usually via clustering (Kohonen, 1990; Salzberg et al., 1995). Prototype positioning learns positions of a reduced training set by optimizing an appropriate objective. The techniques that are most similar to SCC include Decaestecker (1997), Bermejo & Cabestany (1999) who maximize the likelihood of a Gaussian mixture, as well as the recently proposed Stochastic Neighbor Compression (SNC) (Kusner et al., 2014), who use a stochastic neighbor-
Work towards speeding up test-time classification for $k$NN on covariance-valued data is somewhat limited. The JBLD divergence is proposed to speed up individual distance computations. Cherian et al. (2011) show that it is possible to adapt Bregman Ball Trees (BBTs), a generalization of the Euclidean ball tree to Bregman divergences, to the JBLD divergence. This is done using a clever iterative $K$-means method in Cherian et al. (2011) followed by a leaf node projection technique onto relevant Bregman balls. Both of these techniques are complementary to our dataset compression method. Finally, Bucilua et al. (2006) may have been the first to study model compression for machine learning algorithms. In contrast to our work they compress neural networks, which may become more relevant as deep learning becomes increasingly fashionable within the vision community.

6. Conclusion

In many classification settings the sheer amount of distance computations has previously prohibited the use of $k$NN for covariance features. In this paper we showed that data sets of covariance features can be compressed to a small fraction of their original sizes while maintaining their descriptive properties. This drastically speeds up nearest neighbor search and has the potential to make covariance features a powerful alternative to more established image representations, such as bag of visual words. As future work we plan to combine SCC with geodesic metric learning along the Riemannian manifold of SPD matrices.

References

Angiulli, F. Fast condensed nearest neighbor rule. In ICML, pp. 25–32, 2005.

Bermejo, S. and Cabestany, J. Adaptive soft k-nearest-neighbor classifiers. Pattern Recognition, 32:2077–2979, 1999.

Bucilua, Cristian, Caruana, Rich, and Niculescu-Mizil, Alexandru. Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 535–541. ACM, 2006.

Canny, John. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, (6):679–698, 1986.

Cayton, Lawrence. Fast nearest neighbor retrieval for bregman divergences. In Proceedings of the 25th international conference on Machine learning, pp. 112–119. ACM, 2008.

Cherian, Anoop and Sra, Suvrit. Riemannian sparse coding for positive definite matrices. In Computer Vision–ECCV 2014, pp. 299–314. Springer, 2014.

Cherian, Anoop, Sra, Suvrit, Banerjee, Arindam, and Papankikolopoulos, Nikolaos. Efficient similarity search for covariance matrices via the jensen-bregman logdet divergence. In Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 2399–2406. IEEE, 2011.

Coppersmith, Don and Winograd, Shmuel. Matrix multiplication via arithmetic progressions. Journal of symbolic computation, 9(3):251–280, 1990.

Cover, T. and Hart, P. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1):21–27, 1967.
Decaestecker, C. Finding prototypes for nearest neighbour classification by means of gradient descent and deterministic annealing. *Pattern Recognition*, 30(2):281–288, 1997.

Devi, V.S. and Murty, M.N. An incremental prototype set building technique. *Pattern Recognition*, 35(2):505–513, 2002.

Gates, G.W. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18:431–433, 1972.

Goh, Alvina and Vidal, René. Clustering and dimensionality reduction on riemannian manifolds. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–7. IEEE, 2008.

Goldberger, J., Hinton, G.E., Roweis, S.T., and Salakhutdinov, R. Neighbourhood components analysis. In *NIPS*, pp. 513–520. 2004.

Harandi, Mehrtash T, Sanderson, Conrad, Hartley, Richard, and Lovell, Brian C. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *Computer Vision–ECCV 2012*, pp. 216–229. Springer, 2012.

Hart, P.E. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

Hinton, G.E. and Roweis, S.T. Stochastic neighbor embedding. In *NIPS*, pp. 833–840. MIT Press, 2002.

Jayasumana, Sadeep, Hartley, Richard, Salzmann, Mathieu, Li, Hongdong, and Harandi, Mehrtash. Kernel methods on the riemannian manifold of symmetric positive definite matrices. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 73–80. IEEE, 2013.

Kohonen, T. Improved versions of learning vector quantization. In *IJCNN*, pp. 545–550. IEEE, 1990.

Kusner, Matt, Tyree, Stephen, Weinberger, Kilian Q, and Agrawal, Kunal. Stochastic neighbor compression. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 622–630, 2014.

Lai, Kevin, Bo, Lifeng, Ren, Xiaofeng, and Fox, Dieter. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1817–1824. IEEE, 2011.

Laws, Kenneth I. Rapid texture identification. In *24th Annual Technical Symposium*, pp. 376–381. International Society for Optics and Photonics, 1980.

Lazebnik, Svetlana, Schmid, Cordelia, and Ponce, Jean. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pp. 2169–2178. IEEE, 2006.

Oliva, Aude and Torralba, Antonio. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

Pang, Yanwei, Yuan, Yuan, and Li, Xuelong. Gabor-based region covariance matrices for face recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(7):989–993, 2008.

Pennec, Xavier, Fillard, Pierre, and Ayache, Nicholas. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

Porikli, Fatih, Tuzel, Oncel, and Meer, Peter. Covariance tracking using model update based on lie algebra. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pp. 728–735. IEEE, 2006.

Salzberg, S., Delcher, A.L., Heath, D., and Kasif, S. Best-case results for nearest-neighbor learning. *PAMI*, 17(6):599–608, 1995.

Shechtman, Eli and Irani, Michal. Matching local self-similarities across images and videos. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pp. 1–8. IEEE, 2007.

Sra, Suivrit. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *Advances in Neural Information Processing Systems*, pp. 144–152, 2012.

Tomek, I. Two modifications of cnn. *IEEE Trans. on Systems, Man, and Cybernetics, (11):769–772, 1976.

Toussaint, Godfried T. Proximity graphs for nearest neighbor decision rules: recent progress. *Interface*, 34, 2002.

Tuzel, Oncel, Porikli, Fatih, and Meer, Peter. Region covariance: A fast descriptor for detection and classification. In *Computer Vision–ECCV 2006*, pp. 589–600. Springer, 2006.

Tuzel, Oncel, Porikli, Fatih, and Meer, Peter. Pedestrian detection via classification on riemannian manifolds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(10):1713–1727, 2008.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.

Vemulapalli, Raviteja. Riemannian metric learning for symmetric positive definite matrices, 2013. URL [http://bit.ly/2iFLz28](http://bit.ly/2iFLz28).

Wang, Liwei, Li, Yin, Jia, Jiaya, Sun, Jian, Wipf, David, and Rehg, James M. Learning sparse covariance patterns for natural scenes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2767–2774. IEEE, 2012.

Weinberger, K.Q. and Saul, L.K. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

Xiao, Jianxiong, Hays, James, Ehinger, Krista A, Oliva, Aude, and Torralba, Antonio. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pp. 3485–3492. IEEE, 2010.