Deep Tensor Encodings

B Sengupta∗
Dept. of Engineering, University of Cambridge
Trumpington Street
Cambridge, United Kingdom CB2 1PZ
bs573@cam.ac.uk

Y Qian†
Cortexica Vision Systems Limited
Capital Tower – 91 Waterloo Road
London, United Kingdom SE1 8RT
yu.qian@cortexica.com

ABSTRACT
Learning an encoding of feature vectors in terms of an over-complete
dictionary or a information geometric (Fisher vectors) construct is
wide-spread in statistical signal processing and computer vision. In
content based information retrieval using deep-learning classifiers,
such encodings are learnt on the flattened last layer, without adher-
tance to the multi-linear structure of the underlying feature tensor.
We illustrate a variety of feature encodings incl. sparse dictionary
coding and Fisher vectors along with proposing that a structured
tensor factorization scheme enables us to perform retrieval that can
be at par, in terms of average precision, with Fisher vector encoded
image signatures. In short, we illustrate how structural constraints
increase retrieval fidelity.

CCS CONCEPTS
• Theory of computation → Pattern matching; • Computing methodologies → Neural networks;

KEYWORDS
computer vision, deep learning, image retrieval, tensor factoriza-
tion, sparse coding

ACM Reference format:
DOI: arXiv:1703.06324

1 INTRODUCTION
The success of deep-learning lies in constructing feature spaces
where in competing classes of objects, sounds, etc. can be shattered
using high-dimensional hyperplanes. The classifier relies on the
accumulation of representation in the final convolution layer of
a deep neural network. Often times, the classifier performance
increases as one incorporates information from earlier layers of
the neural network. Such a structural constraint has been imposed
on certain deep learning architectures via the inception module. In
addition to decreasing the computational effort, utilization of 1 × 1
convolution filters enables the dimensionality of feature map to be
immensely reduced; in tandem with pooling, the dimensionality
reduces even further. Thus, information from the previous layer(s)
be accumulated and concatenated in the inception module. By
learning the weights feeding to the inception module there is the
additional flexibility in vetting the different sources of information
that reaches deeper layers.

A big demerit of deep-learning architectures is their inability
to perform well in the advent of small amounts of training data.
Tricks such as input (rotation, blur, etc.) and feature augmentation
(in terms of inception module) have proven useful [8]. Such struc-
tural constraints regularize the optimization problem, reducing
over-fitting. In this paper, we propose a much simpler structural
constraint i.e., to utilize the multi-linear structure of deep feature
 tensors. We will first emphasize the importance of feature encoding,
starting with Fisher encoding (of the last layer) followed by a
sparse dictionary based on the last feature layer; this feeds to deep
tensor factorization that brings together tensor learning and deep
learning – cementing the idea that taking into account the high
dimensional multi-linear representation increases the fidelity of
our learnt representation. Albeit, these algorithms are evaluated on
a content-based-image-retrieval (CBIR) setting for a texture dataset,
many of them have been combined in Cortexica’s findSimilar tech-
nology (https://www.cortexica.com/technology/; Figures 2 and 3),
that facilitate retailers to recommend items from fashion databases
comprising inventory items such as tops, trousers, handbags, etc.

2 METHODS
2.1 Dataset and deep-feature generation
In this paper, Describable Textures Dataset (DTD) [4] is used to
evaluate feature encodings for image retrieval. A wide variety of
fashion inventory rely on capturing the differences between vary-
ing textures. Hence, our feature encoding comparison leverages
the DTD dataset, a widely used dataset for texture discrimination.
Rather than recognition and description of object, texture images
in DTD are collected from wild images (Google and Flickr) and
classified based on human visual perception [16], such as direction-
ality (line-like), regularity (polka-dotted and chequered), etc. DTD
contains 5640 wild texture images with 47 describable attributes
drawn from the psychology literature, and is publicly available on
the web at http://www.roberts.ox.ac.uk/vgg/data/dtd/.
Textures can be described via orderless pooling of filter bank response [7]. In deep convolutional neural network (dCNN), the convolutional layers are akin to non-linear filter banks; these have in fact been proved to be better for texture descriptions [5]. Here, the local deep features are extracted from last convolutional layer of a pre-trained VGG-M [3]. This is represented by $T = \{t_1, \ldots, t_N : t \in \mathbb{R}^D\}$, the size of last convolutional layer is $H \times W \times D$, where $D$ denotes the dimension of filter response at the $i$th pixel of last convolution layer; $N = H \times W$ is the total number of local features. Different feature encoding strategies are then utilized for encoding local descriptors. A similarity metric is then applied to rank images. We use the $l_2$ norm (Frobenius norm for matrices and tensors) between vectors as a notion of distance between the query and the database images.

We will now introduce five different encodings of the feature matrix – (a) Fisher encoding, (b) Sparse matrix dictionary learning, (c) t-SVD factorization, (d) Low-rank plus Sparse factorization, and (e) Multilinear Principal Component Analysis (MPCA).

**Feature encoding**

2.1.1 Fisher encoding. We use a Gaussian Mixture Model (GMM) for encoding a probabilistic visual vocabulary on the training dataset. Images are then represented as Fisher Vectors [14, 17] – derivatives of log-likelihood of the model with respect to its parameters (the score function). Fisher encoding describes how the distribution of features of an individual image differs from the distribution fitted to the feature of all training images.

First, a set of $D$ dimension local deep features are extracted from an image and denoted as $T = \{t_1, \ldots, t_N : t \in \mathbb{R}^D\}$. As Ref. [5, 15], a K-component GMM with diagonal covariance is generated on the training set with the parameters $\Theta = (\omega_k, \mu_k, \Sigma_k)_{k=1}^K$, only the derivatives with respect to the mean $|\mu_k|_{k=1}$ and variances $|\Sigma_k|_{k=1}$ are encoded and concatenated to represent an image $T(X, \Theta) = \left\{\frac{\partial L}{\partial \mu_k}, \ldots, \frac{\partial L}{\partial \mu_k}, \frac{\partial L}{\partial \Sigma_k}, \ldots, \frac{\partial L}{\partial \Sigma_k}\right\}$, where,

$$L(\Theta) = \sum_{i=1}^{N} \log(\pi(t_i))$$

$$\pi(t_i) = \sum_{k=1}^{K} \omega_k N(t_i; \mu_k, \Sigma_k) \quad (1)$$

For each component $k$, mean and covariance deviation on each vector dimension $j = 1, 2 \ldots D$ are

$$\frac{\partial L}{\partial \mu_{kj}} = \frac{1}{N \sqrt{\omega_k}} \sum_{i=1}^{N} q_{ik} t_{ij} - \mu_{kj} \sigma_{kj}$$

$$\frac{\partial L}{\partial \Sigma_{kj}} = \frac{1}{N \sqrt{2 \omega_k}} \sum_{i=1}^{N} q_{ik} \left(\frac{t_{ij} - \mu_{kj}}{\sigma_{kj}}\right)^2 - 1 \quad (2)$$

where $q_{ik}$ is the soft assignment weight of feature $t_i$ to the $k$th Gaussian and defined as

$$q_{ik} = \exp \left[ -\frac{1}{2} (t_i - \mu_k)^T \Sigma_k^{-1} (t_i - \mu_k) \right] \sum_{l=1}^{K} \exp \left[ -\frac{1}{2} (t_i - \mu_l)^T \Sigma_l^{-1} (t_i - \mu_l) \right] \quad (3)$$

Just as the image representation, the dimension of Fisher vector is $2KD$, $K$ is the number of components in the GMM, and $D$ is the dimension of local feature descriptor. After $l_2$ normalization on Fisher vector, the Euclidean distance is calculated to find similarity between two images.

2.1.2 Sparse coding on deep features. The compressed (sparse) sensing framework allows us to learn a set of over-complete bases $D$ and sparse weights $\phi$ such that the feature matrix $T$ can be represented by a linear combination of these basis vectors:

$$\arg \min_{D, \phi} \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{2} \| T - D \cdot \phi \|_F^2 + \lambda \| \phi \|_1 \right) \quad (4)$$

The k-SVD algorithm [9] comprises of two stages – first, a sparse coding stage (either using matching pursuit or basis pursuit) and second, a dictionary update stage. In the first stage when the dictionary $D$ is fixed, we solve for $\phi$ using orthogonal matching pursuit (OMP). Briefly, OMP recovers the support of the weights $\phi$ using an iterative greedy algorithm that selects at each step the column of $D$ that is most correlated with the current residual. Practically, we initialize the residual $r_k$, subsequently computing the column that reduces the residual the most $j^*$ and finally adding this column to the support $l_k$

$$r_k = T - D \phi_{k-1}$$

$$j^* = \arg \min_{j=1, \ldots, n} \| r_k - d_j \phi_n \|_2$$

$$l_k = I_{k-1} \cup j^* \quad (5)$$

Iterating through these equations for a pre-specified number of iteration, we can update the sparse weight $\phi$. After obtaining the sparse weights, we use a dictionary update stage where we update only one column of $D$ each time. The update for the $k$-th column is,

$$\frac{\| T - D \cdot \phi \|_F^2}{E_k} \quad (6)$$

$$\| T - \sum_{j=1}^{D} D_j \cdot \phi_j^T \|_F^2$$

$$\| T - \sum_{j \in E_k} D_j \cdot \phi_j^T \|_F^2$$

In order to minimize $\| E_k - D_k \cdot \phi_k^T \|_F^2$ we decompose (SVD) $E_k$ as $UWV^T$. Using this decomposition we utilize a rank-1 approximation to form $d_k = u_0$ and $\phi_k = v_0^T$. We can then iterate this for every column of $D$. Sparsity is attained by collapsing each $\phi_k$ only to non-zero entries, and constraining $E_k$ to the corresponding columns.

For image retrieval, each local deep feature can be encoded by sparse weights $\phi$. Such an image signature can be represented by max pooling of a set of $\phi$, followed by measuring a distance between such sets.
Deep Tensor Encodings

2.1.3 Tensor factorization of deep features. In the earlier section, we relied on an alternate minimization of the dictionary and the loadings (weights) to yield a convex optimization problem. Specifically, the last convolutional layer was used to form a dictionary without reference to the tensorial (multi-linear) representation of feature spaces obtained from the deep convolutional network. Thus, our goal is to approximate these tensors as a sparse conic combinations of the atoms that have been learnt from a dictionary comprising the entire training set. In other words, we would like to obtain an over-complete dictionary such that each feature tensor can be represented as a weighted sum of a small subset of the atoms (loadings) of the dictionary.

There are two decompositions that are used for factorizing tensors, one is based on Tucker decomposition whilst the second is known as Canonical Decomposition/Canonical Polyadic Decomposition (CANDECOMP/CPD), also known as Parallel Factor Analysis (PARAFAC). In the former, tensors are represented by sparse core factors with block structures. Specifically, \( \mathbf{T} \) is approximated as a multi-linear transformation of a small “core” tensor \( \mathbf{G} \) by factor matrices \( \mathbf{A} \) [11],

\[
\mathbf{T} = \mathbf{G} \ast (\mathbf{A}^{(1)} \ast \mathbf{A}^{(2)} \ldots \mathbf{A}^{(N)}) \triangleq \left[ \mathbf{G}^{(i)} ; \mathbf{A}^{(1)} ; \mathbf{A}^{(2)} \ldots \mathbf{A}^{(N)} \right] \quad (7)
\]

In the latter, a tensor \( \mathbf{T} \) is written as a sum of \( R \) rank-1 tensors, each of which can be written as the outer product of \( N \) factor variables i.e.,

\[
\mathbf{T} \approx \sum_{r=1}^{R} \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \ldots \mathbf{a}^{(N)} \triangleq \left[ \mathbf{A}^{(1)} ; \mathbf{A}^{(2)} \ldots \mathbf{A}^{(N)} \right] \quad (8)
\]

It is canonical when the rank \( R \) is minimal; such a decomposition is unique under certain conditions [6]. Even then, due to numerical errors, factorization of a feature matrix obtained using a deep neural network results in a non-unique factorization. Thus, CPD proves inadequate for unique feature encoding. We therefore utilize a factorization that is similar to a 2D-PCA albeit lifted for multi-linear objects. Specifically, we use t-SVD [10] to factorize the feature matrices.

Based on t-SVD. The t-product between two tensors, \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \),

\[
\mathcal{T}_1 \ast \mathcal{T}_2 = \text{fold}\left(\text{circ}(\mathcal{T}_1) \ast \text{unfold}(\mathcal{T}_2)\right)
\]

where, \( \text{circ}() \) creates a block circulant matrix and the unfold operator matricizes the tensor on the tube (3rd) axis. \( \mathbf{S} \) is a f-diagonal tensor that contains the eigen-tuples of the covariance tensor on its diagonal whereas, the columns of \( \mathbf{U} \) are the eigenmatrices of the covariance tensor. The images in the training set are inserted as the second index of the tensor. In other words, using t-SVD, we obtain an orthogonal factor dictionary of the entire training set. Ultimately, a projection of the mean-removed input tensor (i.e., a single feature tensor, \( \mathcal{T}_{\text{test}} \)) on the orthogonal projector \( (\mathbf{U}^T \ast \mathcal{T}_{\text{test}}) \) forms the tensor encoding of each image. The Frobenius norm then measures the closeness between a pair of images (individual tensor projections). Computation efficiency is guaranteed since the comparison between the test image and the database is measured in Fourier domain – the t-product utilizes a fast Fourier transform algorithm in its core [13].

Another feature encoding that we consider is the partition of each individual tensor i.e., \( \mathcal{T} = \mathcal{L} + \mathcal{P} \) where, \( \mathcal{L} \) is a low-rank tensor and \( \mathcal{P} \) is a sparse tensor. We have \( \mathcal{L} = \mathbf{U}_{l,r} \ast \mathbf{S}_{l,r} \ast \mathbf{V}_{l,r}^T \) and \( \mathcal{P} = \mathcal{T} - \mathcal{L} \). \( r \) denotes the truncation index of the singular components.

Based on mPCA. For high-order tensor analysis, multilinear Principal Component Analysis (mPCA) or High Order Singular Value Decomposition (HOSVD) [12, 18] compute a set of orthonormal matrices associated with each mode of a tensor – this is analogous to the orthonormal row and column space of a matrix computed by the matrix PCA/SVD.

In a Tucker decomposition (Eqn. 7) if the factor matrices are constrained to be orthogonal the input tensor can be decomposed as a linear combination of rank-1 tensors. Given a set of \( N \)-order training tensor \( \mathcal{T} \), the objective of mPCA is to find \( N \) linear projection matrices that maximize the total tensor scatter (variance) in the projection subspace. When factor matrices \( \mathbf{A}^{(N)} \) in Eqn. 7 are orthogonal then \( \|\mathcal{T}\|_F^2 = \|\mathbf{G}\|_F^2 \). Each query (test) tensor \( \mathcal{T}_q \) can then be projected using \( \mathbf{Y} = \mathcal{T}_q \ast (\mathbf{A}^{(1)} \ast \mathbf{A}^{(2)} \ast \ldots \mathbf{A}^{(N)})^T \) where the bold-faced matrices represent a low-dimensional space. The objective then becomes learning a set of matrices \( \mathbf{A} \) to admit

\[
\arg \max_{\mathbf{A}^{(1)} \ldots \mathbf{A}^{(N)}} \sum_{m=1}^{M} \|\mathbf{Y}_{m,\text{train}} - \mathbf{Y}\|_F^2 \quad (10)
\]

\( \mathbf{Y} \) is the mean tensor. Since the projection to an \( N \)th order tensor subspace consists of \( N \) projections to \( N \) vector subspaces, in Ref. [12], optimization of the \( N \) projection matrices can be solved by finding each \( \mathbf{A}^{(n)} \) that maximizes the scatter in the \( n \)-mode vector subspace with all other \( A \)-matrices fixed. The local optimization procedure can be iterated until convergence.

EXPERIMENTS

In this section, five deep feature encoding methods are evaluated for image retrieval on the DTD dataset. Images are resized to same size (256 x 256), deep feature is extracted from last convolutional layer of a pre-trained VGG-M. For Fisher vector and sparse coding, deep features are flattened as a set of 1D local features. For t-SVD, deep features are represented as 2D feature maps, and treated as [HxW,1,D] data structures (see Methods). After encoding and \( l_2 \) normalization, the Euclidean distance is calculated to find similarity between two images.

To evaluate image retrieval, Mean Average Precision (MAP) on top 10 rankings are calculated. Two images per category i.e., a total of 94 images are selected as queries from the test dataset. The dataset retrieved includes 3760 images from DTD training and validation datasets. MAP on DTD is listed in Table 1. An example of the retrieval obtained with each method is shown in Figure 2. On each case 10 images are displayed. Top left image is the query used. The rest of images are arranged by similarity to query image as obtained with each encoding method.

Image retrieval amounts to searching large databases to return ranked list of images that have similar properties to that of the
Tensor encodings of deep weights) whereas the tensor dictionaries (t-SVD or mPCA) use efficient encoding for retrieval, having no encoding grossly reduces the average precision. Taking the multi-linear properties of the feature tensor improves retrieval, and the fidelity is at par with Fisher encoding. We further show that computing such multi-linear representation of the feature tensor is computationally much efficient than constructing a sparse dictionary or learning a probabilistic model.

The sparse dictionary encoding has the highest average precision due to the added flexibility of learning the weights as well as imposing the structural constraint of sparsity. Fisher vector encoding has the second highest precision because of its ability to capture the information geometry of the underlying probabilistic model. Specifically, the Fisher tensor encodes the underlying Riemannian metric which augments the encoding with the curvature of the underlying distribution. The multi-linear approaches based on t-SVD, mPCA and low-rank decomposition perform at par with Fisher vectors as they encode the third and higher order interaction in the feature tensors. Comparison of the compute time tells us that amongst all of the methods, encoding deep feature tensors using mPCA is the most time-efficient algorithm.

Our results have not exploited the geometry exhibited by the tensors, for example, one can calculate lagged covariance tensors from the feature tensor – these tensors exhibit a Riemannian geometry due to their positive definite structure. Therefore building a dictionary of co-variance tensors using t-SVD, wherein an Augmented Lagrangian alternating direction method can be employed to learn a sparse representation, is the next viable step to our work. We anticipate that such a multi-linear overcomplete dictionary should at the very least have increased precision to that of the Fisher encoding method. In so far, the last convolutional layer was used to form a dictionary without reference to the earlier layers of the deep neural network. In fact a step forward would be to constrain the construction of an image signature with tensorial information from an earlier layer. The tensor methods rely on factorizing tensors. Comparison of the compute time tells us that amongst all they encode the third and higher order interaction in the feature tensors, so far the methods, encoding deep feature tensors using mPCA is the most time-efficient algorithm.

As shown in Table 1, computing mPCA tensor encodings are computationally efficient in contrast to sparse dictionaries or learning a probabilistic model for Fisher vector encodings. Combined with reasonable retrieval performance, tensor encodings of deep neural features make them a contender for commercial infrastructures.

CONCLUSION

Feature encoding is crucial for comparing images (or videos, text, etc.) that are similar to one another. Our experiments show that whilst sparse encoding of a feature tensor proves to be the most efficient encoding for retrieval, having no encoding grossly reduces the computation time. For large tensors, especially those that emerge from deep neural networks, the tensor methods rely on factorizing tensors. Comparison of the compute time tells us that amongst all

<table>
<thead>
<tr>
<th>Methods</th>
<th>top-1</th>
<th>top-5</th>
<th>top-10</th>
<th>time taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fisher Vector</td>
<td>0.56</td>
<td>0.52</td>
<td>0.48</td>
<td>12ms</td>
</tr>
<tr>
<td>Sparse Coding</td>
<td>0.62</td>
<td>0.49</td>
<td>0.44</td>
<td>35ms</td>
</tr>
<tr>
<td>t-SVD dictionary</td>
<td>0.53</td>
<td>0.42</td>
<td>0.38</td>
<td>2186ms</td>
</tr>
<tr>
<td>mPCA dictionary</td>
<td>0.53</td>
<td>0.43</td>
<td>0.39</td>
<td>5ms</td>
</tr>
<tr>
<td>Low Rank</td>
<td>0.51</td>
<td>0.42</td>
<td>0.38</td>
<td>967ms</td>
</tr>
<tr>
<td>Raw Tensor</td>
<td>0.41</td>
<td>0.35</td>
<td>0.31</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Retrieved results on DTD: a) Fisher vector b) Sparse coding c) t-SVD d) mPCA e) Low rank tensor f) Raw tensor

REFERENCES

A TENSOR NORMS

Let \( t_{ijk} \) be the elements of tensor \( T \), then the Frobenius norm is
\[
\|T\|_F = \|\text{vec}(T)\|_2 = \sqrt{\sum_i \sum_j \sum_k t_{ijk}^2}.
\]
The nuclear (trace) norm is defined as
\[
\|T\|_* = \text{trace}(\sqrt{T^T T}) = \sum_{i=1}^{\min\{m,n\}} \sigma_i.
\]
\( \sigma_i \) are the singular values of \( T \).

B CORTEXICA’S FINDSIMILAR TECHNOLOGY

Figure 2: The findSimilar technology: A consumer takes a photograph of a clothing item. Using proprietary versions of feature encodings that leverage deep-learning as well as multi-scale analysis, the retailer presents similar items from the database.
Figure 3: Feature encodings: Each image is encoded using a (proprietary) combination of encodings described in this paper, along with other patented descriptors. Shown here are examples wherein the query is the top-left item and a ranked list is returned based on similarity.