Sparse Image Coding using a 3D Non-negative Tensor Factorization

Tamir Hazan Simon Polak Amnon Shashua
School of Engineering and Computer Science,
The Hebrew University,
Jerusalem 91904, Israel

Abstract

We introduce an algorithm for a non-negative 3D tensor factorization for the purpose of establishing a local parts feature decomposition from an object class of images. In the past such a decomposition was obtained using non-negative matrix factorization (NMF) where images were vectorized before being factored by NMF. A tensor factorization (NTF) on the other hand preserves the 2D representations of images and provides a unique factorization (unlike NMF which is not unique). The resulting "factors" from the NTF factorization are both sparse (like with NMF) but also separable allowing efficient convolution with the test image. Results show a superior decomposition to what an NMF can provide on all fronts — degree of sparsity, lack of ghost residue due to invariant parts and efficiency of coding of around an order of magnitude better. Experiments on using the local parts decomposition for face detection using SVM and Adaboost classifiers demonstrate that the recovered features are discriminatory and highly effective for classification.

1. Introduction

Finding the optimal collection of filters that capture the "essence" of an object class of images in the most concise and efficiently computable form is a crucial task in visual representation and visual recognition. The literature can be roughly divided into two families of approaches: the first is about efficient filter design which on one hand is rich in their span of variability and on the other hand can be efficiently convolved with an image. Efficiency is typically measured by the number of operations per output pixel in a convolution task. For a filter of a size $r \times s$ the worst efficiency would be the product of dimensions $r \cdot s$, a separable filter will have an efficiency of $r + s$ and there are filter designs with an efficiency of $O(1)$ operations per output pixel ([24, 8, 11], and references therein). It is then a matter of choosing a subset of those filters that are the most "relevant" for the object class, i.e., provide the most accurate classification scores for a given number of filter responses. For example, [24] use a family of horizontal and vertical bar-type filter family and choose the most relevant ones in an incremental (greedy) fashion using Adaboost [**?**].

The second approach does not have a pre-defined filter set and instead treats the desired filters as a problem of finding a low-dimensional basis representation to the training images of the desired object class. The basis vectors are the output filters, whereas the success of the approach lies on applying the right factorization (high dimensional to low dimensional mapping). Probably the most well known example of this approach is Principal Component Analysis (PCA), in which the goal is to find a set of mutually orthogonal basis vectors that capture the directions of maximum variance in the data. In computer vision PCA has been used for the representation and recognition of faces [21, 22, 2], recognition of 3D objects under varying pose [15], tracking of deformable objects [4] and for representations of 3D range data of heads [1].

Higher-order (tensor) decompositions, treating the training images as a 3D cube, have been also proposed where the idea that preserving the 2D form of images is necessary for preserving the spatial coherency of the individual images (something that is lost when images are vectorized in a PCA approach). Those techniques are based on preserving some features of Singular Value Decomposition (SVD) such as to guarantee a reduction to SVD when the image cube is reduced at the limit to copies of a single image [19] or to enforce certain orthogonality constraints (also known as High-Order-SVD) among the basis vectors [13, 23, 26, 10].

The PCA and HOSVD techniques tend to generate filters (basis elements) which have a "holistic" form, i.e., the energy is spread throughout the filter. Techniques which encourage a sparse structure, i.e., the basis images (filters) come out sparse and any image of the class is represented in terms of a small number of basis images out of a large set have been proposed [16] and closely related to that is the work on Independent Component Analysis (ICA) by [7, 3].

A sparse representation has also been achieved using a non-negative factorization (NMF) of the matrix V whose columns are the vectorized training images. The factorization process seeks a decomposition $V = WH$, $W \geq 0, H \geq 0$ where the number of columns k of W are much

smaller than the number of images. The columns of W form the new basis vectors and due to the non-negativity constraint both the basis vectors and the mixing coefficients (columns of H) tend to come out sparse [17, 14]. In particular, [14] introduce a simple and effective iterative technique for performing the decomposition.

A sparse decomposition is appealing for a number of reasons. First, the filters represent local parts of the image set which is consistent with certain theories in visual recognition and with psychological and physiological evidence that support part-based representations in the brain. From a computational standpoint, the convolution with a sparse filter can be done much more efficiently than with a non-sparse filter of the same size.

In this paper we propose an alternative sparse decomposition based on multilinear algebra which we claim *is the natural way to perform a sparse image coding* and which has a significantly higher efficiency and representation power than NMF.

1.1. NTF versus NMF

In the context of achieving a part-based factorization of an image set, the question that naturally arises is what factorization principle would support a decomposition of a collection of training images of a class of objects into a basis of local parts?

There are three drawbacks to the NMF approach — and these are remedied by taking a 3-valence tensor factorization approach instead. The first drawback is that images are not vectors, i.e., vectorizing an image will undoubtedly lead to information loss as the local image structure (i.e., spatial redundancy) would be lost [19]. The second drawback has to do with the general *non-uniqueness* of the NMF strategy. Even if there is an underlying generative model of local parts, there is no guarantee that (even in a perfect fit) the NMF solution would recover it. In other words, It is clear that as a *generative model* the NMF approach makes sense, namely one can imagine simple image settings where the scene is composed out of canonical parts in a variety of positions where these are represented by the columns of W and each image is generated by superposing some of those parts (each part is present or not present in the generated image). What is less clear is whether the NMF process will yield the underlying generative parts (even when there is a perfect fit $V = WH$)? in general this is not true. This point was addressed by [9] where they came up with a set of "rules" that would guarantee a unique decomposition, but the set of rules does not include the common situation of invariant parts which in fact create "ghosts" in the factors and contaminate the sparsity of the basis vectors (see also [6]).

A non-negative tensor factorization (NTF) strategy will represent the input image collection as a 3-way array. A rank-k factorization would correspond to a collection of k rank-1 matrices (the basis images) and mixture coefficients required for generating the original image collection as non-negative super-positions of the basis matrices. The same generative logic applies here as well where the difference lies in the fact that a tensor factorization *is unique* (even without the non-negative constraint) — see next section — and that images are not vectorized in the preparation process, i.e., spatial image structure remains intact. We will return to these points in the experimental section.

A third item is *efficiency*. The factors generated by NTF are not only sparse but also *separable* (rank-1 matrices). It was demonstrated in the past by [19] that the compression ratio of a tensor representation of the image set is an order of a magnitude better compared to a matrix factorization where the vectorized images form the columns of the input matrix. In other words, the number of factors generated by NMF would be comparable to the number of factors generated by NTF for achieving the same reconstruction error (we will demonstrate this in Section 3) yet the NTF factors are separable and are significantly more compressed than the NMF factors.

1.2. What is Known about Tensor Factorizations?

The concept of matrix rank extends quite naturally to higher dimensions: An n-valence tensor G of dimensions $[d_1] \times ... \times [d_n]$ is indexed by n indices $i_1, ..., i_n$ with $1 \leq i_j \leq d_j$ is of rank *at most* k if can be expressed as a sum of k rank-1 tensors, i.e. a sum of n-fold outer-products: $G = \sum_{j=1}^{k} \mathbf{u}_1^j \otimes \mathbf{u}_2^j \otimes ... \otimes \mathbf{u}_n^j$, where $\mathbf{u}_i^j \in R^{d_i}$. The rank of G is the smallest k for which such a decomposition exists. By setting $n = 2$ we obtain the familiar definition of matrix rank which is the smallest number of rank-1 matrices $\mathbf{u}_j \mathbf{v}_j^\top$ whose sum $\sum_j \mathbf{u}_j \mathbf{v}_j^\top$ is equal to G.

Despite sharing the same definition, there are a number of striking differences between the cases $n = 2$ (matrix) and $n > 2$ (tensor). While the rank of a matrix can be found in polynomial time using the SVD algorithm, the rank of a tensor is an NP-hard problem. Even worse, with matrices there is a fundamental relationship between rank-1 and rank-k approximations due to the Eckart-Young theorem where it is sufficient to iterate the process of finding the closest rank-1 matrix to G, subtract it from G and then fit the residue with another rank-1 matrix. This process is repeated until k rank-1 matrices are found — therefore, for matrices the rank-k approximation can be reduced to rank-1 approximation problems. This is not true with tensors in general, i.e., repeatedly subtracting the dominant rank-1 tensor is not a converging process, but only under special cases of orthogonally decomposable tensors (see [27]).

Another striking difference, this time in favor of tensor ranks, is that unlike matrix factorization, which is gener-

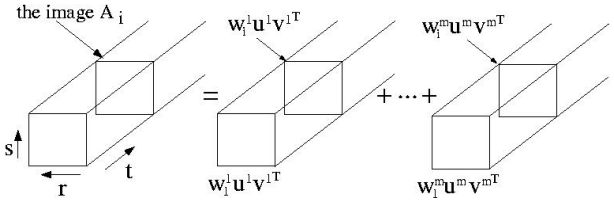


Figure 1: *Representation of the image set as a 3-way array and its rank-k factorization as a sum of k rank-1 tensors $\mathbf{u}^j \otimes \mathbf{v}^j \otimes \mathbf{w}^j$. The j'th rank-1 tensor is made up of slices along the t axis where the i'th slice is a multiple of $\mathbf{u}^j \mathbf{v}^{j\top}$ with the scale equal to w_i^j. The slices of the input tensor G along the t axis are the images $A_1, ..., A_{d_3}$. Therefore, each image A_i is expressed as a superposition of the rank-1 matrices $\mathbf{u}^j \mathbf{v}^{j\top}$.*

ally non-unique for any rank greater than one, a 3-valence tensor decomposition is essentially unique under mild conditions [12] and the situation actually improves in higher dimensions $n > 3$ [20]. The uniqueness property (and this before we introduce non-negativity constraints) is crucial for the sparse coding application mentioned in the previous section as an NMF is not generally unique. An NTF on the other hand would have a direct association between the goodness of fit of the approximate rank-k decomposition and the closeness of obtaining the underlying generative model of the data — and invariant parts are less likely to create ghost patterns in the decomposition.

The body of literature on low-rank decomposition of high-dimensional arrays is mostly focused on special cases where the decomposition is orthogonal, whereas in this paper we are interested in the general multi-linear factorization with *non-negative* entries. A recent attempt to perform an NTF was made by [25] who introduced a heuristic iterative update rule which lacked a convergence proof. Their scheme was based on flattening the tensor into a matrix representation rather than working directly with the outer-products.

2. Algorithms for Low Rank NTF

Let A_t, $t = 1, ..., d_3$ be images of dimensions $d_1 \times d_2$ stacked together as slices of a $d_1 \times d_2 \times d_3$ tensor G whose entries are $G_{r,s,t}$, where $r = 1, ..., d_1; s = 1, ..., d_2$ and $t = 1, ..., d_3$. We wish to factor G into a sum of k rank-1 tensors $G = \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m$, i.e., $G_{r,s,t} = \sum_{m=1}^{k} u_r^m v_s^m w_t^m$ — see Fig. 1. We will begin by describing a positive-preserving gradient decent scheme on the vectors $\{\mathbf{u}^m, \mathbf{v}^m, \mathbf{w}^m\}_{m=1}^k$, such that the sum of squares difference between the elements of the tensor G and the rank-k tensor $\sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m$ is minimized. The positive preserving steps are an extension of the update rule introduced by [14]. In Section 2.2 we will relate the recovered vectors to the way the individual images A_t are represented as a superposition of factors (rank-1 matrices). We consider the following least-squares problem:

$$\min_{\mathbf{u}^m, \mathbf{v}^m, \mathbf{w}^m} \frac{1}{2} \|G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m\|_F^2$$
$$\text{subject to} : \mathbf{u}^m, \mathbf{v}^m, \mathbf{w}^m \geq 0,$$

where $\|A\|_F^2$ is the square Frobenious norm, i.e., the sum of squares of all entries of the tensor elements $A_{r,s,t}$ and $\mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m$ stands for the three fold outer-product. We will be using a gradient decent scheme with a mixture of Jacobi and Gauss-Seidel update scheme and a positive-preserving update rule. Let $< A, B >$ denote the inner-product operation, i.e., $\sum_{r,s,t} A_{r,s,t} B_{r,s,t}$. It is well known that the differential commutes with inner products, i.e., $d < A, A >= 2 < A, dA >$, hence:

$$\frac{1}{2} d < G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m, G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m >$$
$$=< G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m, d \left[G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m \right] >$$

Taking the differential with respect to $\mathbf{u}^j$ and noting that

$$d \left[G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m \right] = -d(\mathbf{u}^j) \otimes \mathbf{v}^j \otimes \mathbf{w}^j,$$

the differential becomes:

$$\begin{aligned} df(\mathbf{u}^j) &= < \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m \, , \, d(\mathbf{u}^j) \otimes \mathbf{v}^j \otimes \mathbf{w}^j > \\ &\quad - < G \, , \, d(\mathbf{u}^j) \otimes \mathbf{v}^j \otimes \mathbf{w}^j > \end{aligned}$$

The differential with respect to the i'th coordinate u_i^j is:

$$\begin{aligned} df(u_i^j) &= < \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m \, , \, \mathbf{e}_i \otimes \mathbf{v}^j \otimes \mathbf{w}^j > \\ &\quad - < G \, , \, \mathbf{e}_i \otimes \mathbf{v}^j \otimes \mathbf{w}^j > \end{aligned}$$

where $\mathbf{e}_i$ is the i'th column of the $d_1 \times d_1$ identity matrix. Using the identity $< \mathbf{x}_1 \otimes \mathbf{y}_1 \, , \, \mathbf{x}_2 \otimes \mathbf{y}_2 >=< \mathbf{x}_1, \mathbf{x}_2 >< \mathbf{y}_1, \mathbf{y}_2 >$ we obtain the partial derivative:

$$\frac{\partial f}{\partial u_i^j} = \sum_{m=1}^{k} u_i^m < \mathbf{v}^m, \mathbf{v}^j >< \mathbf{w}^m, \mathbf{w}^j > - \sum_{s,t} G_{i,s,t} v_s^j w_t^j$$

We will be using a multiplicative update rule by setting the constant $\mu(u_i^j)$ of the gradient descent formula $u_i^j \leftarrow u_i^j - \mu(u_i^j) \frac{\partial f}{\partial u_i^j}$ to be:

$$\mu(u_i^j) = \frac{u_i^j}{\sum_{m=1}^{k} u_i^m < \mathbf{v}^m, \mathbf{v}^j >< \mathbf{w}^m, \mathbf{w}^j >} \qquad (1)$$

thereby obtaining the following update rule:

$$u_i^j \leftarrow \frac{u_i^j \sum_{s,t} G_{i,s,t} v_s^j w_t^j}{\sum_{m=1}^k u_i^m < \mathbf{v}^m, \mathbf{v}^j >< \mathbf{w}^m, \mathbf{w}^j >} \quad (2)$$

Likewise, the update rules for v_i^j and w_i^j are as follows:

$$v_i^j \leftarrow \frac{v_i^j \sum_{r,t} G_{r,i,t} u_r^j w_t^j}{\sum_{m=1}^k v_i^m < \mathbf{u}^m, \mathbf{u}^j >< \mathbf{w}^m, \mathbf{w}^j >} \quad (3)$$

$$w_i^j \leftarrow \frac{w_i^j \sum_{r,s} G_{r,s,i} u_r^j v_s^j}{\sum_{m=1}^k w_i^m < \mathbf{u}^m, \mathbf{u}^j >< \mathbf{v}^m, \mathbf{v}^j >} \quad (4)$$

Note that the update rule preserves non-negativity provided that the initial guess for the vectors $\mathbf{u}^m, \mathbf{v}^m, \mathbf{w}^m$ are non-negative. In iteration (t) of the update process, the values of $\mathbf{u}^j$ are updated Jacobi style with respect to the entries u_i^j for $i = 1, ..., d_1$, and are updated Gauss-Seidel style with respect to the entries of other vectors $\{\mathbf{u}^m\}_{m \neq j}$ and the vectors $\{\mathbf{v}^m, \mathbf{w}^m\}_{m=1}^k$.

In general the convergence proof of the multiplicative rule was introduced by [14] for the bilinear case. The main difference is that our update rule is performed in Gauss-Seidel fashion for the vectors $\mathbf{u}^1, ..., \mathbf{u}^k$ while their update rule is performed Jacobi style. Since we use Jacobi-type update rule only for a single vector $\mathbf{u}^j$ the optimization function with respect to the variables u_i^j has a diagonal Hessian matrix — the proof of this is below.

Proposition 1 *For every $1 \leq j \leq k$ the d_1-variate function $f(\mathbf{u}^j) = \frac{1}{2}\|G - \sum_{m=1}^k \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m\|_F^2$ is quadratic, convex and its Hessian is $c_j I$ where I is the $d_1 \times d_1$ identity matrix and $c_j =< \mathbf{v}^j, \mathbf{v}^j >< \mathbf{w}^j, \mathbf{w}^j >= \|\mathbf{v}^j\|^2 \|\mathbf{w}^j\|^2$.*

Proof: The first derivatives are:

$$\frac{\partial f}{\partial u_i^j} = \sum_{m=1}^k u_i^m < \mathbf{v}^m, \mathbf{v}^j >< \mathbf{w}^m, \mathbf{w}^j > - \sum_{s,t} G_{i,s,t} v_s^j w_t^j$$

Therefore we conclude that

$$\frac{\partial^2 f}{\partial u_i^j \partial u_i^j} =< \mathbf{v}^j, \mathbf{v}^j >< \mathbf{w}^j, \mathbf{w}^j >$$

and

$$\frac{\partial^2 f}{\partial u_i^j \partial u_k^j} = 0 \quad \text{for} \quad i \neq k$$

Since the Hessian is positive definite and constant, it follows that the function is convex and quadratic. □

We show below that the multiplicative update rule with respect to the variables u_i^j reduces the optimization function. First we will show that the gradient step size is less than the inverse ratio of the Hessian diagonal value. Then we will prove that this step size is suitable for our optimization.

Proposition 2

$$\mu(u_i^j) \;<\; 1/\frac{\partial^2 f}{\partial u_i^j \partial u_i^j} = \frac{1}{< \mathbf{v}^j, \mathbf{v}^j >< \mathbf{w}^j, \mathbf{w}^j >}$$

Proof:

$$\begin{aligned} \mu(u_i^j) &= \frac{u_i^j}{\sum_{m=1}^k u_i^m < \mathbf{v}^m, \mathbf{v}^j >< \mathbf{w}^m, \mathbf{w}^j >} \\ &< \frac{u_i^j}{u_i^j < \mathbf{v}^j, \mathbf{v}^j >< \mathbf{w}^j, \mathbf{w}^j >} \end{aligned}$$

where the inequality holds since all the elements are positive and by reducing positive elements in the denominator we increase the fraction value. □

To complete the convergence proof we need to show that the step size $\mu(u_i^j) = \mu$ along the gradient reduces the optimization function. The general statement and its proof is below:

Proposition 3 *Let $f(x_1, ..., x_n)$ be a quadratic function to the real numbers with Hessian of the form $H = cI$ where $c > 0$. Given a point $x = (x_1^t, ..., x_n^t) \in R^n$ and a point $x^{t+1} = x^t - \mu(\bigtriangledown f(x^t))$, and a decent step $0 < \mu < \frac{1}{c}$ then $f(x^{t+1}) < f(x^t)$*

Proof: By Fourier expansion of $f(x + y)$ we get

$$f(x + y) = f(x) + \bigtriangledown f(x)^\top y + \frac{1}{2} y^\top H y$$

Choosing $x = x^t$ and $y = -\mu \bigtriangledown f(x^t)$ we get

$$\begin{aligned} f(x^t - \mu \bigtriangledown f(x^t)) &= f(x^t) - \mu(\bigtriangledown f(x^t)^\top \bigtriangledown f(x^t)) \\ &\quad + \frac{1}{2}\mu^2 c(\bigtriangledown f(x^t)^\top \bigtriangledown f(x^t)) \end{aligned}$$

We need to show that $f(x^t) - f(x^{t+1}) > 0$:

$$\begin{aligned} f(x^t) - f(x^{t+1}) &= \mu \| \bigtriangledown f(x^t)\|^2 - \frac{1}{2}\mu^2 c \| \bigtriangledown f(x^t)\|^2 \\ &= \mu \| \bigtriangledown f(x^t)\|^2 (1 - \frac{1}{2} c\mu) \end{aligned}$$

The result follows since $\mu < \frac{1}{c}$. □

Following the same mathematical reasoning we can prove the convergence of the computational updates for the vectors $\{\mathbf{v}^m, \mathbf{w}^m\}_{m=1}^k$. The derivation of the general n-way array non-negative decomposition can be found in [18].

2.1. NTF under Relative Entropy

Following [14] we also describe positive-preserving update rule for the relative entropy cost function. Given a tensor G we consider the best positive rank-k approximation for

$$D(G|| \sum_{m=1}^k \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m)$$

where $D(A||B) = \sum_{r,s,t}(A_{r,s,t} log \frac{A_{r,s,t}}{B_{r,s,t}} - A_{r,s,t} + B_{r,s,t})$. Let $< A, B >$ denote the inner product operation and let $log(A)_{r,s,t} = log(A_{r,s,t})$ then $D(A||B) =< A, log(A) > - < A, log(B) > - < \mathbf{1}, A > + < \mathbf{1}, B >$:

$$d\ D(G||\sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m) =$$

$$< \mathbf{1}, d\sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m > - < G, d\, log(\sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m) >$$

Taking the differential with respect to $\mathbf{u}^j$ we obtain:

$$df(\mathbf{u}_i^j) =< \mathbf{1}, d(\mathbf{u}^j) \otimes \mathbf{v}^j \otimes \mathbf{w}^j > - < G, \frac{d(\mathbf{u}^j) \otimes \mathbf{v}^j \otimes \mathbf{w}^j)}{\sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m} >$$

where division is coordinate-wise. The partial derivatives are:

$$\frac{\partial f}{\partial u_i^j} = \sum_{s,t} v_s^j w_t^j - \sum_{s,t} G_{i,s,t} \frac{v_s^j w_t^j}{\sum_{m=1}^{k} u_i^m v_s^m w_t^m}$$

Choosing a gradient decent step of size $\mu(u_i^j) = \frac{u_i^j}{\sum_{s,t} v_s^j w_t^j}$ results in the positive preserving update rule:

$$u_i^j \leftarrow u_i^j \frac{\sum_{s,t} G_{i,s,t} \frac{v_s^j w_t^j}{\sum_{m=1}^{k} u_i^m v_s^m w_t^m}}{\sum_{s,t} v_s^j w_t^j}$$

The convergence proof is omitted due to lack of space.

2.2. Extracting the Factors from the Factorization

The update rules eqn. 2,3,4 will converge to a local minima of the energy function $\|G - \sum_{m=1}^{k} \mathbf{u}^m \otimes \mathbf{v}^m \otimes \mathbf{w}^m\|^2$ with non-negative entries. The original images $A_1, ..., A_{d_3}$ make up the slices of G along the t coordinate. The relationship between the 2D images and the rank-1 tensor factorization is captured by the set of rank-1 matrices $\tau_j = \mathbf{u}^j \mathbf{v}^{j^\top}$ such that each A_t is represented by a superposition of $\tau_1, ..., \tau_k$ with the mixture coefficients taken from $\mathbf{w}^1, ..., \mathbf{w}^k$ — this is derived below using the Khatri-Rao product notation.

Let $U = [\mathbf{u}^1, ..., \mathbf{u}^k]$ be a $d_1 \times k$ matrix, $V = [\mathbf{v}^1, ..., \mathbf{v}^k]$ of dimension $d_2 \times k$, and $W = [\mathbf{w}^1, ..., \mathbf{w}^k]$ of dimension $d_3 \times k$. The Khatri-Rao product of two matrices $U \odot V$ is defined as the $d_1 d_2 \times k$ matrix $[\mathbf{u}^1 \otimes \mathbf{v}^1, ..., \mathbf{u}^k \otimes \mathbf{v}^k]$. Let $(U \odot V)W^\top = [X_1, ..., X_{d_3}]$, then each column X_t is $vec(A_t)$ the vector representation (column-wise concatenation) of the image A_t. In other words, each vectorized image $vec(A_t)$ is a linear combination of the $\mathbf{u}^j \otimes \mathbf{v}^j = vec(\mathbf{u}^j \mathbf{v}^{j^\top})$ with coefficients taken from the t'th row of W. In matrix form we have that $A_t = U \Lambda_t V^\top$ where $\Lambda_t = diag(w_t^1, ..., w_t^k)$.

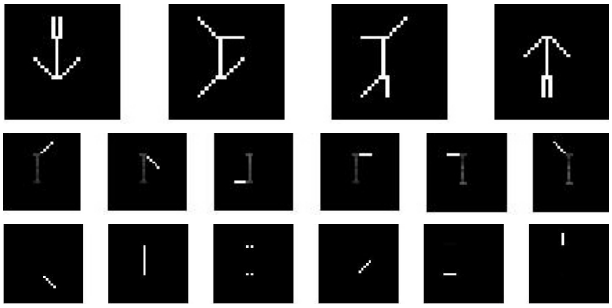


Figure 2: *Comparing the factors generated by NMF (middle row) and NTF (bottom row) from a set of 256 images of the Swimmer library (sample in top row). The NMF factors contains ghosts of invariant parts (the torso) which contaminate the sparse representation.*

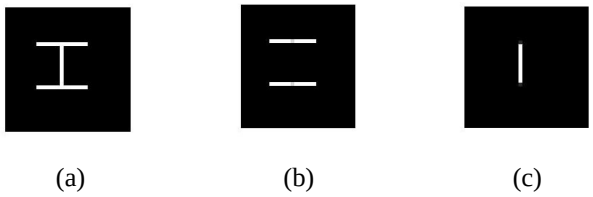
To conclude, the result of the NTF procedure are the rank-1 factors $\tau_j = \mathbf{u}^j \mathbf{v}^{j^\top}$ which form a basis (filters) for representing the object class of images. The vectors $\mathbf{w}^1, ..., \mathbf{w}^k$ can be discarded.

3. Experiments

We start with empirical verification to the effect of decomposition uniqueness of NTF compared to NMF and its effect on the success of recreating the underlying generative model. Following [9] we built the Swimmer image set of 256 images of dimensions 32×32. Each image contains a "torso" (the invariant part) of 12 pixels in the center and four "limbs" of 6 pixels that can be in one of 4 positions — see Fig. 2 for examples. The NMF scheme of [14] for finding 17 factors running over the image set correctly resolves the local parts but fails on the torso. The torso being an invariant part as it appears in the same position through the entire set appears as a "ghost" in all the factors. The NTF on the other hand, contains a unique factorization and correctly resolves all the 17 parts. The number of rank-1 factors is 50 (since the diagonal limbs are not rank-1 parts). The rank-1 matrices corresponding to the limbs are superimposed in the display in Fig. 2 for purposes of clarity.

For another illustration of the power of NTF, consider the problem of resolving local parts from a *single* image. In an NMF framework this cannot be achieved as a single image, even if copied multiple times, would still be decomposed into itself. With NTF on the other hand, we copied the single image 20 times and run the NTF on the image cube. The experiment was conducted on one of the swimmer images Fig 3a and on a real face image Fig 3d. With respect to Fig 3a an NTF algorithm recovered the 2 factors which reconstruct the image. With respect to Fig 3d the factors were grouped together and shown in Fig 3(e-h) demonstrating local part decomposition.

For another illustration of the power of NTF compared



(a) (b) (c)

(d) (e) (f) (g) (h)

Figure 3: *Running NTF on a single image copied 20 times to form a 3D cube. Upper row: (a) the original image, (b),(c) the two recovered factors. Lower row: (d) the original image, (e)-(h) the recovered factors in 4 groups.*

	linear	poly $d = 5$	RBF
NTF (50)	91.9%	95.3%	95.9%
NMF (50)	91.6%	94%	95%
NMF (20)	87.5%	90.1%	89%
NMF (6)	83.2%	84.3%	86%
PCA	90.8%	94%	91.7%

Figure 5: *Using the filter responses of NMF, NTF, PCA as measurements for an SVM classifier, with linear, polynomial of degree five and RBF kernels, trained over the MIT CBCL face dataset. 50 NTF factors were used compared to 50, 20 and 6 NMF facotrs in three separate experiments. The percentages correct over the test set are displayed in the table. The NTF outperformed the NMF even when 50 NMF factors were used (20-fold higher space than NTF).*

to NMF, we have applied NMF and NTF to the set of 2429, 19×19, face images from the MIT CBCL database. Fig. 4 shows the leading factors generated by NMF - one can clearly see ghost structures and the part decomposition is complicated (an observation supported by empirical studies done by other groups such as on Iris image sets in [6]). The NTF factors (rank-1 matrices) have a sharper decomposition into sparse components. We also grouped together factors whose energy are localized in the same image region and took their sum. The sum of factors represent a higher rank part decomposition which is useful for purposes of getting a better idea what face structures are deemed as "parts" in the decomposition. One can clearly see the parts corresponding to eyes, cheeks, shoulders, etc.

Another consequence of representing the image set as a 3D tensor is that the spatial redundancy is factored in the decomposition (which is not the case when the images are vectorized as in the NMF framework) — therefore one should expect a more efficient representation (higher compression rate). We computed 50 factors with NTF and used them to reconstruct the original images. Each NMF factor is a full rank image and is thus comparable to 19 NTF factors in terms of space requirements. We compared the fidelity of the NTF reconstruction with 50 factors to the NMF reconstruction with 4 factors. One can clearly see a striking difference in the quality of reconstruction which validates the increased coding efficiency of the tensor representation of the image set compared to a 2D representation. We then used 50 NMF factors and obtained a similar quality reconstruction to the NTF with the same number of factors (a 20-fold reduction in space).

The next experiment, shown in Fig. 5, used the filter responses as measurements for a Support Vector Machine (SVM) classifier [5]. We used the MIT CBCL face set to recover the factors. The measurement vector representing an image was the inner-product between the factors and the input image. Those measurement vectors over positive (faces) and negative (non-faces) examples were fed into the SVM classifier. We varied the kernel of the SVM from linear to polynomial of degree five to RBF and recorded the percentage correct over a test set. The training and testing was conducted on a "leave one out" paradigm where $4/5$ of the set was used for training and the remaining $1/5$ of the set was used for testing. Each trial a different training and testing subsets were used and the results were averaged over the trials. We used 50 NTF factors and 50, 20 and 6 NMF factors in three separate trials and also used PCA factors for comparison. The measurements induced from the NTF factors generated the highest classification accuracy compared to 50 NMF factors which contain a 20-fold space increase, i.e., despite the much higher compression rate of the NTF compared to NMF and PCA the resulting local features apparently better captured the face set.

For Another illustration of the power of NTF, we constructed from the filters weak learners for Adaboost. The factors were recovered from the MIT CBCL face database in the following way: We used 200 NTF rank-1 factors grouped to 90 local parts, to form a low rank part based representation. We created 100 NMF factors (those contain a 10-fold space increase compared to the NTF factors). We also computed 100 PCA factors for comparison. The main idea of the Adaboost is to assign to each example of the training set a weight. At the beginning all the weights are equal, but in every round the weak learner returns a hypothesis, and the weights of all examples classified wrong by that hypothesis are increased. That way the weak learner is enforced to focus on the difficult examples of the training set. The final hypothesis is a combination of the hypothesis of all rounds, namely a weighted majority vote, where the hypothesis with lower classification error have

Figure 4: *NMF versus NTF on face images. Top to Bottom: leading factors of NMF, leading factors of NTF, summed factors of NTF located in the same region (resulting in higher rank factors), reconstruction using 50 NTF factors, reconstruction using 4 NMF factors, reconstruction using 50 NMF facotrs, and original images for comparison.*

	Adaboost
NTF (200)	90.9%
NMF (100)	84.1%
PCA (100)	82.8%

Figure 6: *Using the filters as weak learners for an Adaboost classifier, trained over the MIT CBCL face database. 200 NTF factors were used, grouped to 90 low rank weak learners and compared to 100 NMF factors. The percentage correct over the test set is shown in the table. The NTF is superior to the NMF although it used 10-fold less space than the NMF*

higher weight. The results shown in fig. 6 demonstrate significantly higher accuracy rate when the NTF-based weak learners were used.

References

[1] J.J. Atick, P.A. Griffin, and N.A. Redlich. Statistical approach to shape-from-shading: deriving 3d face surfaces from single 2d images. *Neural Computation*, 1997.

[2] P.N Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. In *Proceedings of the European Conference on Computer Vision*, 1996.

[3] A.J. Bell and T.J. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation 7(6)*, pages 1129–1159, 1995.

[4] Michael J. Black and D. Jepson. Eigen-tracking: Robust matching and tracking of articulated objects using a view-based representation. In *eccv*, pages 329–342, Cambridge, England, 1996.

[5] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifers. In *Proc. of the 5th ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[6] M. Chu, F. Diele, R. Plemmons, and S. Ragni. Optimality, computation and interpretation of nonnegative matrix factorizations. *SIAM Journal on Matrix Analysis*, 2004.

[7] P. Comon. Independent component analysis, a new concept? *Signal processing 36(3)*, pages 11–20, 1994.

[8] F.C. Crow. Summed-area tables for texture mapping. In *Conf. on Comp. Graphics and Interactive Techniques*, pages 207–212, 1984.

[9] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts. In *Proceedings of the conference on Neural Information Processing Systems (NIPS)*, 2003.

[10] R.A. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(84), 1970.

[11] Y. Hel-Or and H. Hel-Or. Real time pattern matching using projection kernels. In *Proceedings of the International Conference on Computer Vision*, pages 1486–1493, Nice, France, 2003.

[12] J.B. Kruksal. Three way arrays: rank and uniqueness of trilinear decomposition, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18:95–138, 1977.

[13] L De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *Matrix Analysis and Application*, 21:1253–1278, 2000.

[14] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[15] H. Murase and S.K. Nayar. Learning and recognition of 3D objects from appearance. In *IEEE 2nd Qualitative Vision Workshop*, pages 39–50, New York, NY, June 1993.

[16] B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(13), 1996.

[17] P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Envirometrics*, 5:111–126, 1994.

[18] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[19] A. Shashua and A. Levin. Linear image coding for regression and classification using the tensor-rank principle. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, Dec. 2001.

[20] N.D. Sidiropoulos and R. Bro. On the uniqueness of multilinear decomposition of n-way arrays. *Journal of Chemometrics*, 14:229–239, 2000.

[21] L. Sirovich and M. Kirby. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4(3):519–524, 1987.

[22] M.Turk and A.Pentland. Eigen faces for recognition. *J. of Cognitive Neuroscience*, 3(1), 1991.

[23] M.A.O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Proceedings of the European Conference on Computer Vision*, pages 447–460, 2002.

[24] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, Dec. 2001.

[25] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22(12):1255–1261, 2001.

[26] L. Xianqian and N.D. Sidiropoulos. Cramer-rao lower boubds for low-rank decomposition of multidimensional arrays. *IEEE Transactions on Signal Processing*, 49(9), 2001.

[27] T. Zhang and G.H Golub. Rank-one approximation to high order tensors. *Matrix Analysis and Applications*, 23:534–550, 2001.