The Incremental Multiresolution Matrix Factorization Algorithm

Vamsi K. Ithapu[†], Risi Kondor[§], Sterling C. Johnson[†], Vikas Singh[†]
[†]University of Wisconsin-Madison, [§]University of Chicago
http://pages.cs.wisc.edu/~vamsi/projects/incmmf.html

Abstract

Multiresolution analysis and matrix factorization are foundational tools in computer vision. In this work, we study the interface between these two distinct topics and obtain techniques to uncover hierarchical block structure in symmetric matrices – an important aspect in the success of many vision problems. Our new algorithm, the incremental multiresolution matrix factorization, uncovers such structure one feature at a time, and hence scales well to large matrices. We describe how this multiscale analysis goes much farther than what a direct "global" factorization of the data can identify. We evaluate the efficacy of the resulting factorizations for relative leveraging within regression tasks using medical imaging data. We also use the factorization on representations learned by popular deep networks, providing evidence of their ability to infer semantic relationships even when they are not explicitly trained to do so. We show that this algorithm can be used as an exploratory tool to improve the network architecture, and within numerous other settings in vision.

1. Introduction

Matrix factorization lies at the heart of a spectrum of computer vision problems. While the wide ranging and extensive use of factorization schemes within structure from motion [38], face recognition [40] and motion segmentation [10] have been known, in the last decade, there is renewed interest in these ideas. Specifically, the celebrated work on low rank matrix completion [6] has enabled deployments in a broad cross-section of vision problems from independent components analysis [18] to dimensionality reduction [42] to online background estimation [43]. Novel extensions based on Robust Principal Components Analysis [13, 6] are being developed each year.

In contrast to factorization methods, a distinct and rich body of work based on early work in signal processing is arguably even more extensively utilized in vision. Specifically, Wavelets [34] and other related ideas (curvelets [5], shearlets [24]) that loosely fall under multiresolution analysis (MRA) based approaches drive an overwhelming majority of techniques within feature extraction [29] and representation learning [34]. Also, Wavelets remain the "go to" tool for image denoising, compression, inpainting, shape analysis and other applications in video processing [30]. SIFT features can be thought of as a special case of the so-called Scattering Transform (using theory of Wavelets) [4]. Remarkably, the "network" perspective of Scattering Transform at least partly explains the invariances being identified by deep representations, further expanding the scope of multiresolution approaches informing vision algorithms.

The foregoing discussion raises the question of whether there are any interesting bridges between Factorization and Wavelets. This line of enquiry has recently been studied for the most common "discrete" object encountered in vision – graphs. Starting from the seminal work on Diffusion Wavelets [11], others have investigated tree-like decompositions on matrices [25], and organizing them using wavelets [16]. While the topic is still nascent (but evolving), these non-trivial results suggest that the confluence of these seemingly distinct topics potentially holds much promise for vision problems [17]. Our focus is to study this interface between Wavelets and Factorization, and demonstrate the immediate set of problems that can potentially benefit. In particular, we describe an efficient (incremental) multiresolution matrix factorization algorithm.

To concretize the argument above, consider a representative example in vision and machine learning where a factorization approach may be deployed. Figure 1 shows a set of covariance matrices computed from the representations learned by AlexNet [23], VGG-S [9] (on some ImageNet classes [35]) and medical imaging data respectively. As a first line of exploration, we may be interested in characterizing the apparent parsimonious "structure" seen in these matrices. We can easily verify that invoking the de facto constructs like sparsity, low-rank or a decaying eigen-spectrum cannot account for the "block" or cluster-like structures inherent in this data. Such block-structured kernels were the original motivation for block low-rank and hierarchical factorizations [36, 8] — but a multiresolution scheme is much more natural — in fact, ideal — if one can decompose the

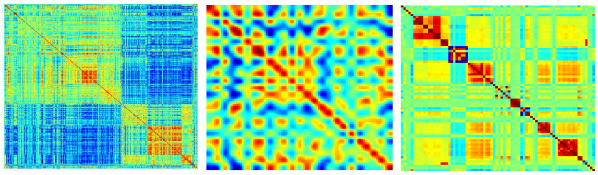


Figure 1. Left-to-right example category (or class) covariances from AlexNet, VGG-S (of a few ImageNet classes) and medical imaging data.

matrix in a way that the blocks automatically 'reveal' themselves at multiple resolutions. Conceptually, this amounts to a sequential factorization while accounting for the fact that each level of this hierarchy must correspond to approximating some non-trivial structure in the matrix. A recent result introduces precisely such a *multiresolution matrix factorization* (MMF) algorithm for symmetric matrices [21].

Consider a symmetric matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$. PCA decomposes $\mathbf{C}$ as $\mathbf{Q}^T \mathbf{\Lambda} \mathbf{Q}$ where $\mathbf{Q}$ is an orthogonal matrix, which, in general, is dense. On the other hand, sparse PCA (sPCA) [46] imposes sparsity on the columns of $\mathbf{Q}$, allowing for fewer dimensions to interact that may not capture global patterns. The factorization resulting from such individual low-rank decompositions cannot capture hierarchical relationships among data dimensions. Instead, MMF applies a sequence of carefully chosen sparse rotations $\mathbf{Q}^1, \mathbf{Q}^2, \dots \mathbf{Q}^L$ to factorize $\mathbf{C}$ in the form

$$\mathbf{C} = (\mathbf{Q}^1)^T (\mathbf{Q}^2)^T \dots (\mathbf{Q}^L)^T \mathbf{\Lambda} \mathbf{Q}^L \dots \mathbf{Q}^2 \mathbf{Q}^1,$$

thereby uncovering soft hierarchical organization of different rows/columns of $\mathbf{C}$. Typically the $\mathbf{Q}^\ell$s are sparse k^{th}-order rotations (orthogonal matrices that are the identity except for at most k of their rows/columns), leading to a hierarchical tree-like matrix organization. MMF was shown to be an efficient compression tool [39] and a preconditioner [21]. Randomized heuristics have been proposed to handle large matrices [22]. Nevertheless, factorization involves searching a combinatorial space of row/column indices, which restricts the order of the rotations to be small (typically, ≤ 3). Not allowing higher order rotations restricts the richness of the allowable block structure, resulting in a hierarchical decomposition that is "too localized" to be sensible or informative (reverting back to the issues with sPCA and other block low-rank approximations).

A fundamental property of MMF is the sequential composition of rotations. In this paper, we exploit the fact that the factorization can be parameterized in terms of an *MMF graph* defined on a sequence of higher-order rotations. Unlike alternate batch-wise approaches [39], we start with a small, randomly chosen block of $\mathbf{C}$, and gradually 'insert' new rows into the factorization – hence we refer to this as an *incremental* MMF. We show that this insertion procedure manipulates the topology of the MMF graph, thereby providing an efficient algorithm for constructing higher order MMFs. Our **contributions** are: **(A)** We present a fast and efficient incremental procedure for constructing higher order (large k) MMFs on large dense matrices; **(B)** We evaluate the efficacy of the higher order factorizations for relative leveraging of sets of pixels/voxels in regression tasks in vision; and **(C)** Using the output structure of incremental MMF, we visualize the semantics of categorical relationships inferred by deep networks, and, in turn, present some exploratory tools to adapt and modify the architectures.

2. Multiresolution Matrix Factorization

Notation: We begin with some notation. Matrices are bold upper case, vectors are bold lower case and scalars are lower case. $[m] := \{1, \dots, m\}$ for any $m \in \mathbb{N}$. Given a matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ and two set of indices $\mathcal{S}_1 = \{r_1, \dots r_k\}$ and $\mathcal{S}_2 = \{c_1, \dots c_p\}$, $\mathbf{C}_{\mathcal{S}_1, \mathcal{S}_2}$ will denote the block of $\mathbf{C}$ cut out by the rows $\mathcal{S}_1$ and columns $\mathcal{S}_2$. $\mathbf{C}_{:,i}$ is the i^{th} column of $\mathbf{C}$. $\mathbf{I}_m$ is the m-dimensional identity. $SO(m)$ is the group of m dimensional orthogonal matrices with unit determinant. $\mathcal{R}^m_{\mathcal{S}}$ is the set of m-dimensional symmetric matrices which are diagonal except for their $\mathcal{S} \times \mathcal{S}$ block ($\mathcal{S}$–core-diagonal matrices).

Multiresolution matrix factorization (MMF), introduced in [21, 22], retains the locality properties of sPCA while also capturing the global interactions provided by the many variants of PCA, by applying not one, but multiple *sparse* rotation matrices to $\mathbf{C}$ in sequence. We have the following.

Definition. Given an appropriate class $\mathcal{O} \subseteq SO(m)$ of sparse rotation matrices, a depth parameter $L \in \mathbb{N}$ and a sequence of integers $m = d_0 \geq d_1 \geq \dots \geq d_L \geq 1$, the **multi-resolution matrix factorization (MMF)** of a symmetric matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ is a factorization of the form

$$\mathcal{M}(\mathbf{C}) := \overline{\mathbf{Q}}^T \mathbf{\Lambda} \overline{\mathbf{Q}} \quad \text{with} \quad \overline{\mathbf{Q}} = \mathbf{Q}^L \dots \mathbf{Q}^2 \mathbf{Q}^1, \qquad (1)$$

where $\mathbf{Q}^\ell \in \mathcal{O}$ and $\mathbf{Q}^\ell_{[m] \setminus \mathcal{S}_{\ell-1}, [m] \setminus \mathcal{S}_{\ell-1}} = \mathbf{I}_{m - d_\ell}$ for some nested sequence of sets $[m] = \mathcal{S}_0 \supseteq \mathcal{S}_1 \supseteq \dots \supseteq \mathcal{S}_L$ with $|\mathcal{S}_\ell| = d_\ell$ and $\mathbf{\Lambda} \in \mathcal{R}^m_{\mathcal{S}_L}$.

$\mathcal{S}_{\ell-1}$ is referred to as the 'active set' at the ℓ^{th} level, since $\mathbf{Q}^\ell$ is identity outside $[m] \setminus \mathcal{S}_{\ell-1}$. The nesting of the $\mathcal{S}_\ell$s implies that after applying $\mathbf{Q}^\ell$ at some level ℓ, $\mathcal{S}_{\ell-1} \setminus \mathcal{S}_\ell$ rows/columns are removed from the active set, and are not operated on subsequently. This active set trimming is done at all L levels, leading to a nested subspace interpretation for the sequence of compressions $\mathbf{C}^\ell = \mathbf{Q}^\ell \mathbf{C}^{\ell-1} (\mathbf{Q}^\ell)^T$ ($\mathbf{C}^0 = \mathbf{C}$ and $\mathbf{\Lambda} = \mathbf{C}^L$). In fact, [21] has shown that, for a general class of symmetric matrices, MMF from Definition 2 entails a Mallat style multiresolution analysis (MRA) [28]. Observe that depending on the choice of $\mathbf{Q}^\ell$, only a few dimensions of $\mathbf{C}^{\ell-1}$ are forced to interact, and so the composition of rotations is hypothesized to extract subtle or softer notions of structure in $\mathbf{C}$.

Since multiresolution is represented as matrix factorization here (see (1)), the $\mathcal{S}_{\ell-1} \setminus \mathcal{S}_\ell$ columns of $\overline{\mathbf{Q}}$ correspond to "wavelets". While $d_1, d_2, \ldots$ can be any monotonically decreasing sequence, we restrict ourselves to the simplest case of $d_\ell = m - \ell$. Within this setting, the number of levels L is at most $m - k + 1$, and each level contributes a single wavelet. Given $\mathcal{S}_1, \mathcal{S}_2, \ldots$ and $\mathcal{O}$, the matrix factorization of (1) reduces to determining the $\mathbf{Q}^\ell$ rotations and the residual $\mathbf{\Lambda}$, which is usually done by minimizing the squared Frobenius norm error

$$\min_{\mathbf{Q}^\ell \in \mathcal{O}, \mathbf{\Lambda} \in \mathcal{R}^m_{S_L}} \|\mathbf{C} - \mathcal{M}(\mathbf{C})\|^2_{\text{Frob}}. \tag{2}$$

The above objective can be decomposed as a sum of contributions from each of the L different levels (see Proposition 1, [21]), which suggests computing the factorization in a greedy manner as $\mathbf{C} = \mathbf{C}^0 \mapsto \mathbf{C}^1 \mapsto \mathbf{C}^2 \mapsto \ldots \mapsto \mathbf{\Lambda}$. This error decomposition is what drives much of the intuition behind our algorithms.

After $\ell - 1$ levels, $\mathbf{C}^{\ell-1}$ is the compression and $\mathcal{S}_{\ell-1}$ is the active set. In the simplest case of $\mathcal{O}$ being the class of so-called k–point rotations (rotations which affect at most k coordinates) and $d_\ell = m - \ell$, at level ℓ the algorithm needs to determine three things: (a) the k–tuple t^ℓ of rows/columns involved in the rotation, (b) the nontrivial part $\mathbf{O} := \mathbf{Q}^\ell_{t^\ell, t^\ell}$ of the rotation matrix, and (c) s^ℓ, the index of the row/column that is subsequently designated a wavelet and removed from the active set. Without loss of generality, let s^ℓ be the last element of t^ℓ. Then the contribution of level ℓ to the squared Frobenius norm error (2) is (see supplement)

$$\begin{aligned} \mathcal{E}(\mathbf{C}^{\ell-1}; \mathbf{O}^\ell; t^\ell, s) &= 2 \sum_{i=1}^{k-1} [\mathbf{O} \mathbf{C}^{\ell-1}_{t^\ell, t^\ell} \mathbf{O}^T]^2_{k,i} \\ &+ 2[\mathbf{O}\mathbf{B}\mathbf{B}^T\mathbf{O}^T]_{k,k} \quad \text{where} \quad \mathbf{B} = \mathbf{C}^{\ell-1}_{t^\ell, \mathcal{S}_{\ell-1} \setminus t^\ell}, \end{aligned} \tag{3}$$

and, in the definition of $\mathbf{B}$, t^ℓ is treated as a set. The factorization then works by minimizing this quantity in a greedy fashion, i.e.,

$$\begin{aligned} \mathbf{Q}^\ell, t^\ell, s^\ell &\leftarrow \operatorname*{argmin}_{\mathbf{O}, t, s} \mathcal{E}(\mathbf{C}^{\ell-1}; \mathbf{O}; t, s) \\ \mathcal{S}_\ell &\leftarrow \mathcal{S}_{\ell-1} \setminus s^\ell \quad ; \quad \mathbf{C}^\ell = \mathbf{Q}^\ell \mathbf{C}^{\ell-1} (\mathbf{Q}^\ell)^T. \end{aligned} \tag{4}$$

3. Incremental MMF

We now motivate our algorithm using (3) and (4). Solving (2) amounts to estimating the L different k-tuples $t^1, \ldots, t^L$ sequentially. At each level, the selection of the best k-tuple is clearly combinatorial, making the exact MMF computation (i.e., explicitly minimizing (2)) very costly even for $k = 3$ or 4 (this has been independently observed in [39]). As discussed in Section 1, higher order MMFs (with large k) are nevertheless inevitable for allowing arbitrary interactions among dimensions (see supplement for a detailed study), and our proposed incremental procedure exploits some interesting properties of the factorization error and other redundancies in k-tuple computation. The core of our proposal is the following setup.

3.1. Overview

Let $\tilde{\mathbf{C}} \in \mathbb{R}^{(m+1)\times(m+1)}$ be the extension of $\mathbf{C}$ by a *single* new column $\mathbf{w} = [\mathbf{u}^T, v]^T$, which manipulates $\mathbf{C}$ as:

$$\tilde{\mathbf{C}} = \left[\begin{array}{c|c} \mathbf{C} & \mathbf{u} \\ \hline \mathbf{u}^T & v \end{array}\right]. \tag{5}$$

The goal is to compute $\mathcal{M}(\tilde{\mathbf{C}})$. Since $\mathbf{C}$ and $\tilde{\mathbf{C}}$ share all but one row/column (see (5)), if we have access to $\mathcal{M}(\mathbf{C})$, one should, in principle, be able to modify $\mathbf{C}$'s underlying sequence of rotations to construct $\mathcal{M}(\tilde{\mathbf{C}})$. This *avoids* having to recompute everything for $\tilde{\mathbf{C}}$ from scratch, i.e., performing the greedy decompositions from (4) on the entire $\tilde{\mathbf{C}}$.

The hypothesis for manipulating $\mathcal{M}(\mathbf{C})$ to compute $\mathcal{M}(\tilde{\mathbf{C}})$ comes from the precise computations involved in the factorization. Recall (3) and the discussion leading up to the expression. At level $\ell + 1$, the factorization picks the 'best' candidate rows/columns from $\mathbf{C}^\ell$ that correlate the most with each other, so that the resulting diagonalization induces the smallest possible off-diagonal error over the rest of the active set. The components contributing towards this error are driven by the inner products $(\mathbf{C}^\ell_{:,i})^T \mathbf{C}^\ell_{:,j}$ for some columns i and j. In some sense, the largest such correlated rows/columns get picked up, and adding one new entry to $\mathbf{C}^\ell_{:,i}$ may not change the *range* of these correlations. Extending this intuition across all levels, we argue that

$$\operatorname*{argmax}_{i,j} \tilde{\mathbf{C}}^T_{:,i} \tilde{\mathbf{C}}_{:,j} \approx \operatorname*{argmax}_{i,j} \mathbf{C}^T_{:,i} \mathbf{C}_{:,j}. \tag{6}$$

Hence, the k-tuples computed from $\mathbf{C}$'s factorization are reasonably good candidates even after introducing $\mathbf{w}$. To better formalize this idea, and in the process present our algorithm, we parameterize the output structure of $\mathcal{M}(\mathbf{C})$ in terms of the sequence of rotations and the wavelets.

3.2. The graph structure of $\mathcal{M}(\mathbf{C})$

If one has access to the sequence of k-tuples $t^1, \ldots, t^L$ involved in the rotations and the corresponding wavelet indices $(s^1, \ldots, s^L)$, then the factorization is straightforward to compute i.e., there is no greedy search anymore. Recall that by definition $s^\ell \in t^\ell$ and $s^\ell \notin \mathcal{S}_\ell$ (see (4)). To that end, for a given $\mathcal{O}$ and L, $\mathcal{M}(\mathbf{C})$ can be 'equivalently' represented using a depth L *MMF graph* $\mathcal{G}(\mathbf{C})$. Each level of this graph shows the k-tuple t^ℓ involved in the rotation, and

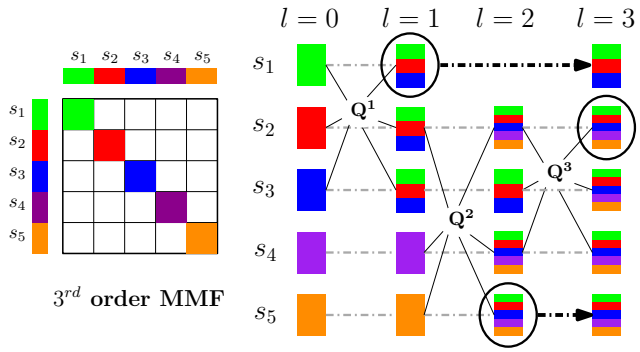


Figure 2. An example 5×5 matrix, and its 3^{rd} order MMF graph (better in color). $\mathbf{Q}^1$, $\mathbf{Q}^2$ and $\mathbf{Q}^3$ are the rotations. s_1, s_5 and s_2 are wavelets (marked as black ellipses) at $l = 1, 2$ and 3 respectively. The arrows imply that a wavelet is not involved in future rotations.

the corresponding wavelet s^ℓ i.e., $\mathcal{G}(\mathbf{C}) := \{t^\ell, s^\ell\}_1^L$. Interpreting the factorization in this way is notationally convenient for presenting the algorithm. More importantly, such an interpretation is central for visualizing hierarchical dependencies among dimensions of $\mathbf{C}$, and will be discussed in detail in Section 4.3. An example of such a 3^{rd} order MMF graph constructed from a 5×5 matrix is shown in Figure 2 (the rows/columns are color coded for better visualization). At level $\ell = 1$, s_1, s_2 and s_3 are diagonalized while designating the rotated s_1 as the wavelet. This process repeats for $\ell = 2$ and 3. As shown by the color-coding of different compositions, MMF gradually teases out *higher-order* correlations that can only be revealed after composing the rows/columns at one or more scales (levels here).

For notational convenience, we denote the MMF graphs of $\mathbf{C}$ and $\tilde{\mathbf{C}}$ as $\mathcal{G} := \{t^\ell, s^\ell\}_1^L$ and $\tilde{\mathcal{G}} := \{\tilde{t}^\ell, \tilde{s}^\ell\}_1^{L+1}$. Recall that $\tilde{\mathcal{G}}$ will have one more level than $\mathcal{G}$ since the row/column $\mathbf{w}$, indexed $m+1$ in $\tilde{\mathbf{C}}$, is being added (see (5)). The goal is to estimate $\tilde{\mathcal{G}}$ without recomputing all the k-tuples using the greedy procedure from (4). This translates to *inserting* the new index $m+1$ into the t^ℓs and modifying s^ℓs accordingly. Following the discussion from Section 3.1, incremental MMF argues that inserting this one new element into the graph will not result in global changes in its topology. Clearly, in the pathological case, $\mathcal{G}$ may change arbitrarily, but as argued earlier (see discussion about (6)) the chance of this happening for non-random matrices with reasonably large k is small. The core operation then is to compare the *new* k-tuples resulting from the addition of $\mathbf{w}$ to the best ones from $[m]^k$ provided via $\mathcal{G}$. If the newer k-tuple gives better error (see (3)), then it will *knock out* an existing k-tuple. This constructive insertion and knock-out procedure is the incremental MMF.

3.3. Inserting a new row/column

The basis for this incremental procedure is that one has access to $\mathcal{G}$ (i.e., MMF on $\mathbf{C}$). We first present the algorithm assuming that this "initialization" is provided, and revisit this aspect shortly. The procedure starts by setting $\tilde{t}^\ell = t^\ell$ and $\tilde{s}^\ell = s^\ell$ for $\ell = 1, \ldots, L$. Let $\mathcal{I}$ be the set of elements (indices) that needs to be inserted into $\mathcal{G}$. At the start (the first level) $\mathcal{I} = \{m+1\}$ corresponding to $\mathbf{w}$. Let $\tilde{t}^1 = \{p_1, \ldots, p_k\}$. The new k-tuples that account for inserting entries of $\mathcal{I}$ are $\{m+1\} \cup t^1 \setminus p_i$ $(i = 1, \ldots, k)$. These new k candidates are the probable alternatives for the existing $\tilde{t}^1$. Once the best among these $k+1$ candidates is chosen, an existing p_i from $\tilde{t}^1$ may be knocked out.

If $\tilde{s}^1$ gets knocked out, then $\mathcal{I} = \{\tilde{s}^1\}$ for future levels. This follows from MMF construction, where wavelets at ℓ^{th} level are not involved in later levels. Since $\tilde{s}^1$ is knocked out, it is the new inserting element according to $\mathcal{G}$. On the other hand, if one of the $k-1$ scaling functions is knocked out, $\mathcal{I}$ is not updated. This simple process is repeated sequentially from $\ell = 1$ to L. At $L+1$, there are no estimates for $\tilde{t}^{L+1}$ and $\tilde{s}^{L+1}$, and so, the procedure simply selects the best k-tuple from the remaining active set $\tilde{\mathcal{S}}_L$. Algorithm 1 summarizes this insertion and knock-out procedure.

Algorithm 1 INSERTROW$(\mathbf{C}, \mathbf{w}, \{t^\ell, s^\ell\}_{\ell=1}^L)$

Output: $\{\tilde{t}^\ell, \tilde{s}^\ell\}_{\ell=1}^{L+1}$
$\tilde{\mathbf{C}}^0 \leftarrow \tilde{\mathbf{C}}$ as in (5)
$z^1 \leftarrow m+1$
for $\ell = 1$ to $L-1$ **do**
  $\{\tilde{t}^\ell, \tilde{s}^\ell, z^{\ell+1}, \mathbf{Q}^\ell\} \leftarrow$ CHECKINSERT$(\tilde{\mathbf{C}}^{\ell-1}; t^\ell, s^\ell, z^\ell)$
  $\tilde{\mathbf{C}}^\ell = \mathbf{Q}^\ell \tilde{\mathbf{C}}^{\ell-1} (\mathbf{Q}^\ell)^T$
end for
$\mathcal{T} \leftarrow$ GENERATETUPLES$([m+1] \setminus \cup_{\ell=1}^{L-1} \tilde{s}^\ell(\tilde{\mathbf{C}}))$
$\{\tilde{\mathbf{O}}, \tilde{t}^L, \tilde{s}^L\} \leftarrow \operatorname{argmin}_{\mathbf{O}, t \in \mathcal{T}, s \in t} \mathcal{E}(\tilde{\mathbf{C}}^{L-1}; \mathbf{O}; t, s)$
$\mathbf{Q}^L = \mathbf{I}_{m+1}$, $\mathbf{Q}^L_{\tilde{t}^L, \tilde{t}^L} = \tilde{\mathbf{O}}$, $\tilde{\mathbf{C}}^L = \mathbf{Q}^L \tilde{\mathbf{C}}^{L-1} (\mathbf{Q}^L)^T$

Algorithm 2 CHECKINSERT$(\mathbf{A}, \hat{t}, \hat{s}, z)$

Output: $\tilde{t}$, $\tilde{s}$, z, $\mathbf{Q}$
$\mathcal{T} \leftarrow$ GENERATETUPLES$(\hat{t}, z)$
$\{\tilde{\mathbf{O}}, \tilde{t}, \tilde{s}\} \leftarrow \operatorname{argmin}_{\mathbf{O}, t \in \mathcal{T}, s \in t} \mathcal{E}(\mathbf{A}; \mathbf{O}; t, s)$
if $\tilde{s} \in z$ **then**
  $z \leftarrow (z \cup \hat{s}) \setminus \tilde{s}$
end if
$\mathbf{Q} = \mathbf{I}_{m+1}$, $\mathbf{Q}_{\tilde{t}, \tilde{t}} = \tilde{\mathbf{O}}$

3.4. Incremental MMF Algorithm

Observe that Algorithm 1 is for the setting from (5) where one extra row/column is added to a given MMF, and clearly, the incremental procedure can be repeated as more and more rows/columns are added. Algorithm 3 summarizes this incremental factorization for arbitrarily large and dense matrices. It has two components: an initialization on some randomly chosen small block (of size $\tilde{m} \times \tilde{m}$) of

the entire matrix $\mathbf{C}$; followed by insertion of the remaining $m-\tilde{m}$ rows/columns using Algorithm 1 in a streaming fashion (similar to $\mathbf{w}$ from (5)). The initialization entails computing a batch-wise MMF on this small block ($\tilde{m} \geq k$).

BATCHMMF: Note that at each level ℓ, the error criterion in (3) can be explicitly minimized via an exhaustive search over all possible k-tuples from $\mathcal{S}_{\ell-1}$ (the active set) and a randomly chosen (using properties of QR decomposition [31]) dictionary of k^{th} order rotations. If the dictionary is large enough, the exhaustive procedure would lead to the smallest possible decomposition error (see (2)). However, it is easy to see that this is combinatorially large, with an overall complexity of $\mathcal{O}(n^k)$ [22] and will not scale well beyond $k=4$ or so. Note from Algorithm 1 that the error criterion $\mathcal{E}(\cdot)$ in this second stage which inserts the rest of the $m-\tilde{m}$ rows is performing an exhaustive search as well.

Algorithm 3 INCREMENTAL MMF($\mathbf{C}$)

Output: $\mathcal{M}(\mathbf{C})$
  $\bar{\mathbf{C}} = \mathbf{C}_{[\tilde{m}],[\tilde{m}]}$, $L = m-k+1$
  $\{t^\ell, s^\ell\}_1^{\tilde{m}-k+1} \leftarrow$ BATCHMMF($\bar{\mathbf{C}}$)
  for $j \in \{\tilde{m}+1, \ldots, m\}$ **do**
    $\{t^\ell, s^\ell\}_1^{j-k+1} \leftarrow$ INSERTROW($\bar{\mathbf{C}}, \mathbf{C}_{j,:}, \{t^\ell, s^\ell\}_1^{j-k}$)
    $\bar{\mathbf{C}} = \mathbf{C}_{[j],[j]}$
  end for
  $\mathcal{M}(\mathbf{C}) := \{t^\ell, s^\ell\}_1^L$

Other Variants: The are two alternatives that avoid this exhaustive search. Since $\mathbf{Q}^\ell$'s job is to diagonalize some k rows/columns (see Definition 2), one can simply pick the relevant $k \times k$ block of $\mathbf{C}^\ell$ and compute the best $\mathbf{O}$ (for a given t^ℓ). Hence the first alternative is to bypass the search over $\mathcal{O}$ (in (4)), and simply use the eigen-vectors of $\mathbf{C}^\ell_{t^\ell,t^\ell}$ for some tuple t^ℓ. Nevertheless, the search over $\mathcal{S}_{\ell-1}$ for t^ℓ still makes this approximation reasonably costly. Instead, the k-tuple selection may be approximated while keeping the exhaustive search over $\mathcal{O}$ intact [22]. Since diagonalization effectively nullifies correlated dimensions, the best k-tuple can be the k rows/columns that are maximally correlated. This is done by choosing some $s_1 \sim \mathcal{S}_{\ell-1}$ (from the current active set), and picking the rest by

$$s_2, \ldots, s_k \leftarrow \underset{s_i \sim \mathcal{S}_{\ell-1} \setminus s_1}{\operatorname{argmin}} \sum_{i=2}^{k} \frac{(\mathbf{C}^{\ell-1}_{:,s_1})^T \mathbf{C}^{\ell-1}_{:,s_i}}{\|\mathbf{C}^{\ell-1}_{:,s_1}\| \|\mathbf{C}^{\ell-1}_{:,s_i}\|} \tag{7}$$

This second heuristic (which is related to (6) from Section 3.1) has been shown to be robust [22], however, for large k it might miss some k-tuples that are vital to the quality of the factorization. Depending on $\tilde{m}$, and the available computational resources at hand, these alternatives can be used instead of the earlier proposed exhaustive procedure for the initialization. Overall, the incremental procedure scales efficiently for very large matrices, compared to using the batch-wise scheme on the entire matrix.

4. Experiments

We study various computer vision and medical imaging scenarios (see supplement for details) to evaluate the quality of incremental MMF factorization and show its utility. We first provide evidence for factorization's efficacy in selecting the relevant features of interest for regression. We then show that the resultant MMF graph is a useful tool for visualizing/decoding the learned task-specific representations.

4.1. Incremental versus Batch MMF

The first set of evaluations compares the incremental MMF to the batch version (including the exhaustive search based and the two approximate variants from Section 3.4). Recall that MMF error is the off-diagonal norm of $\mathbf{\Lambda}$, except for the $\mathcal{S}_L \times \mathcal{S}_L$ block (see (1)), and the smaller the error is, the closer the factorization is to being exact (see 2). We observed that the incremental MMFs incur approximately the same error as the batch versions, while achieving $\gtrsim 20-25$ times speed-up compared to a single-core implementation of the batch MMF. Specifically, across 6 different toy examples and 3 covariance matrices constructed from real data, the loss in factorization error is $\lesssim 4\%$ of $\|\mathbf{C}\|_{\text{Frob}}$, with no strong dependence on the fraction of the initialization $\tilde{m}$ (see Algorithm 3). Due to space restrictions, these simulations are included in the supplement.

4.2. MMF Scores

The objective of MMF (see (2)) is the signal that is not accounted for by the k^{th}-order rotations of MMF (it is 0 whenever $\mathbf{C}$ is *exactly* factorizable). Hence, $\|(\mathbf{C}-\mathcal{M}(\mathbf{C}))_{i,:}\|$ is a measure of the extra information in the i^{th} row that cannot be reproduced by hierarchical compositions of the rest. Such value-of-information summaries, referred to as *MMF scores*, of all the dimensions of $\mathbf{C}$ give an importance sampling distribution, similar to statistical leverage scores [3, 27]. These samplers drive several regression tasks in vision including gesture tracking [33], face alignment/tracking [7] and medical imaging [15]. Moreover, the authors in [27] have shown that statistical leverage type marginal importance samplers may not be optimal for regression. On the other hand, MMF scores give the conditional importance or "relative leverage" of each dimension/feature given the remaining ones. This is because MMF encodes the hierarchical block structure in the covariance, and so, the MMF scores provide better importance samplers than statistical leverages. We first demonstrate this on a large dataset with 80 predictors/features and 1300 instances. Figure 3(a,b) shows the instance covariance matrices after selecting the 'best' 5% of features. The block

structure representing the two classes, diseased and non-diseased, is clearly more apparent with MMF score sampling (see the yellow block vs.the rest in Figure 3(b)).

We exhaustively compared leverages and MMF scores on a medical imaging regression task on region-of-interest (ROI) summaries from positron emission tomography (PET) images. The goal is to predict the cognitive score summary using the imaging ROIs. (see Figure 3(c), and supplement for details). Despite the fact that the features have high degree of block structure (covariance matrix from Figure 3(c)), this information is rarely, if ever, utilized within the downstream models, say multi-linear regression for predicting health status. Here we train a linear model using a *fraction* of these voxel ROIs sampled according to statistical leverages (from [3]) and relative leverage from MMF scores. Note that, unlike LASSO, the feature samplers are agnostic to the responses (a setting similar to optimal experimental design [12]). The second row of Figure 3 shows the Adjusted-R^2 of the resulting linear models, and Figure 3(h,i) show the corresponding F-statistic. The x-axis corresponds to the fraction of the ROIs selected using the leverage (black lines) and MMF scores (red lines). As shown by the red vs. black curves, the voxel ROIs picked by MMF are better both in terms of adjusted-R^2 (the explainable variance of the data) and F-statistic (the overall significance). More importantly, the first few ROIs picked up by MMF scores are more informative than those from leverage scores (left end of x-axis in Figure 3(d-i)). Figure 3(j,k) show the gain in AUC of adjusted-R^2 as the order of MMF changes (x-axis). Clearly the performance gain of MMF scores is large. The error bars in these plots are omitted for clarity (see supplement for details, and other plots/comparisons). These results show that MMF scores can be used within numerous regression tasks where the number of predictors is large with sample sizes.

4.3. MMF graphs

The ability of a feature to succinctly represent the presence of an object/scene is, at least, in part, governed by the relationship of the learned representations across multiple object classes/categories. Beyond object-specific information, such cross-covariate contextual dependencies have shown to improve the performance in object tracking and recognition [45] and medical applications [20] (a motivating aspect of adversarial learning [26]). Visualizing the histogram of gradients (HoG) features is one such interesting result that demonstrates the scenario where a correctly learned representation leads to a false positive [41], for instance, the HoG features of a duck image are similar to a car HoG. [37, 14] have addressed similar aspects for deep representations by visualizing image classification and detection models, and there is recent interest in designing tools for visualizing what the network perceives when predicting a test label [44]. As shown in [1], the contextual images that a deep network (even with good detection power) desires to see may not even correspond to real-world scenarios.

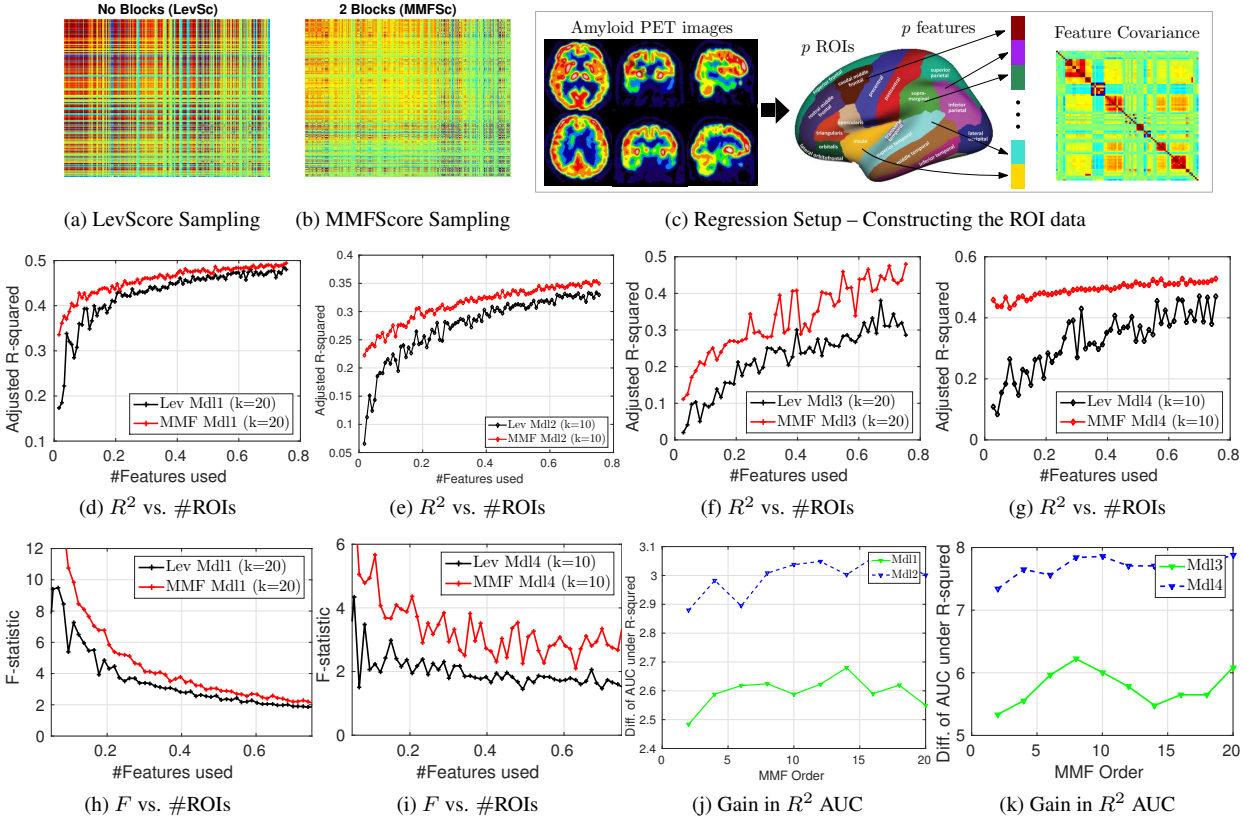
The evidence from these works motivate a simple question – *Do the semantic relationships learned by the deep representations associate with those seen by humans?* For instance, can such models infer that cats are closer to dogs than they are to bears; or that bread goes well with butter/cream rather than, say, salsa. Invariably, addressing these questions amounts to learning hierarchical and categorical relationships in the class-covariance of hidden representations. Using classical techniques may not easily reveal interesting, human-relateable, trends as was shown very recently by [32]. There are at least few reasons, but most importantly, the covariance of hidden representations (in general) has parsimonious structure with multiple compositions of blocks (the left two images in Figure 1 are from AlexNet and VGG-S). As motivated in Section 1, and later described in Section 3.2 using Figure 2, a MMF graph is the natural object to analyze such parsimonious structure.

4.3.1 Decoding the deep

A direct application of MMF on the covariance of hidden representations reveals interesting hierarchical structure about the "perception" of deep networks. To precisely walk through these compositions, consider the last hidden layer (FC7, that feeds into softmax) representations from a VGG-S network [9] corresponding to 12 different ImageNet classes, shown in Figure 4(a). Figure 4(b,c) visualize a 5^{th} order MMF graph learned on this class covariance matrix.

The semantics of breads and sides. The 5^{th} order MMF says that the five categories – *pita*, *limpa*, *chapati*, *chutney* and *bannock* – are most representative of the localized structure in the covariance. Observe that these are four different flour-based main courses, and a side *chutney* that shared strongest context with the images of *chapati* in the training data (similar to the body building and dumbell images from [1]). MMF then picks *salad*, *salsa* and *saute* representations' at the 2^{nd} level, claiming that they relate the strongest to the *composition* of breads and *chutney* from the previous level (see visualization in Figure 4(b,c)). Observe that these are in fact the sides offered/served with bread. Although VGG-S was *not* trained to predict these relations, according to MMF, the representations are inherently learning them anyway – a fascinating aspect of deep networks i.e., they are seeing what humans may infer about these classes.

Any dressing? What are my dessert options? Let us move to the 3^{rd} level in Figure 4(b,c). *margarine* is a cheese based dressing. *shortcake* is dessert-type meal made from *strawberry* (which shows up at 4^{th} level) and bread (the composition from previous levels). That is the full course. The last level corresponds to *ketchup*, which is an outlier, distinct from the rest of the 10 classes – a typical order of



(a) LevScore Sampling (b) MMFScore Sampling (c) Regression Setup – Constructing the ROI data

(d) R^2 vs. #ROIs (e) R^2 vs. #ROIs (f) R^2 vs. #ROIs (g) R^2 vs. #ROIs

(h) F vs. #ROIs (i) F vs. #ROIs (j) Gain in R^2 AUC (k) Gain in R^2 AUC

Figure 3. **Evaluating Feature Importance Sampling of MMF Scores vs. Leverage Scores** (a,b) Visualizing apparent (if any) blocks in instance covariance matrices using best 5% features, (c) Regression setup (see structure in covariance), (d-g) Adjusted R^2, and (h,i) Fstatistic of linear models, (j,k) gains in R^2. Mdl1-Mdl4 are linear models constructed on different datasets (see supplement). $\tilde{m} = 0.1m$ (from Algorithm 3) for these evaluations.

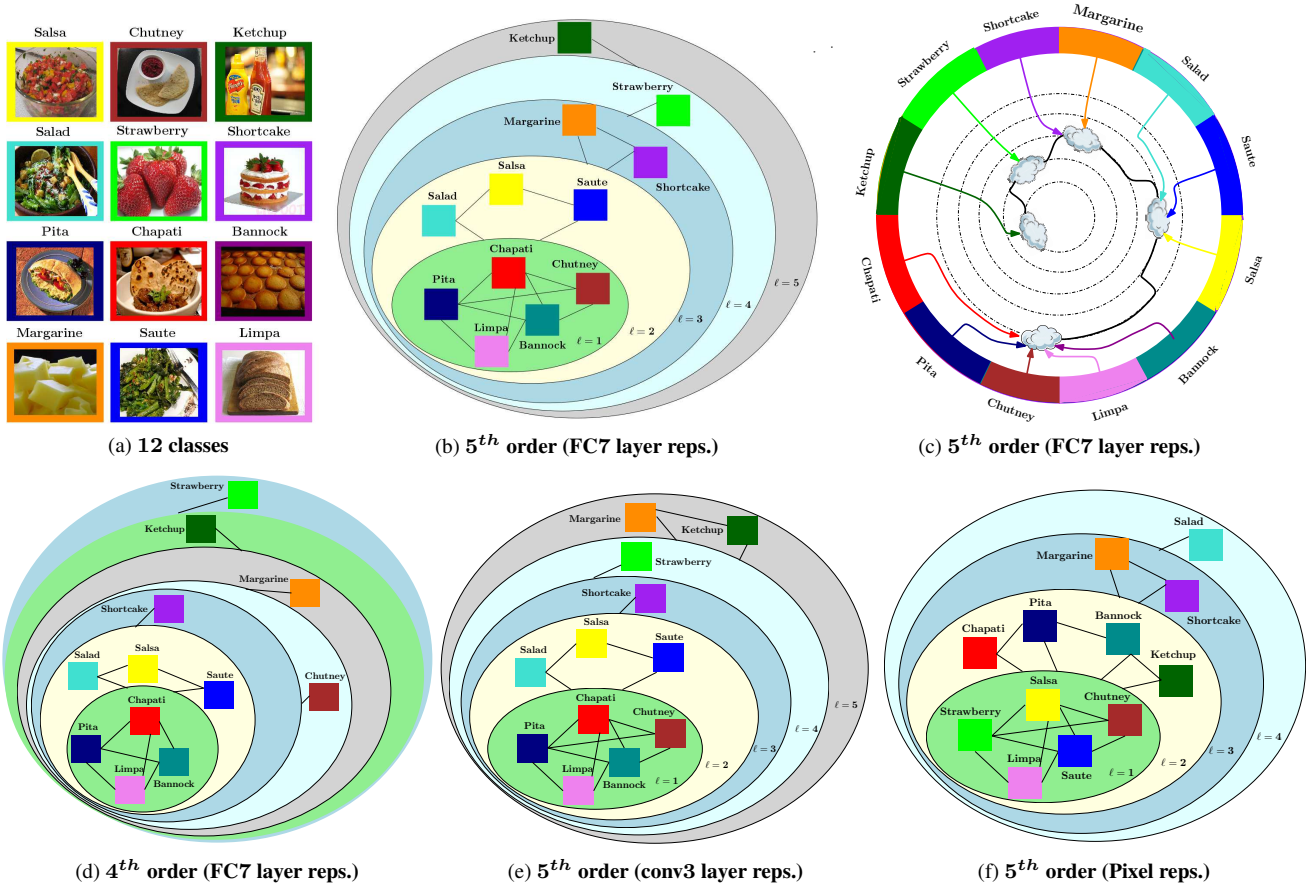
dishes involving the chosen breads and sides does not include hot sauce or ketchup. Although *shortcake* is made up of strawberries, "conditioned" on the 1^{st} and 2^{nd} level dependencies, it is less useful in summarizing the covariance structure. An interesting summary of this hierarchy from Figure 4(b,c) is – an order of *pita* with side *ketchup* or *strawberries* is atypical in the data seen by these networks.

4.3.2 Are we reading tea leaves?

It is reasonable to ask if this description is meaningful since the semantics drawn above are subjective. We provide explanations below. First, the networks are *not* trained to learn the hierarchy of categories – the task was object/class detection. Hence, the relationships are completely a by-product of the power of deep networks to learn contextual information, *and* the ability of MMF to model these compositions by uncovering the structure in the covariance matrix. Supplement provides further evidence by visualizing such hierarchy from few dozens of other ImageNet classes. Second, one may ask if the compositions are sensitive/stable to the order k – a critical hyperparameter of MMF. Figure 4(d) uses a 4^{th} order MMF, and the resulting hierarchy is similar to that from Figure 4(b). Specifically, the different breads and sides show up early, and the most distinct categories (*strawberry* and *ketchup*) appear at the higher levels. Similar patterns are seen for other choices of k (see supplement).

Further, if the class hierarchy in Figures 4(b–d) is non-spurious, then similar trends should be implied by MMF's on different (higher) layers of VGG-S. Figure 4(e) shows the compositions from the 10^{th} layer representations (the outputs from 3^{rd} convolutional layer of VGG-S) of the 12 classes in Figure 4(a). The strongest compositions, the 8 classes from $\ell = 1$ and 2, are already picked up halfway thorough the VGG-S, providing further evidence that the compositional structure implied by MMF is data-driven. We further discuss this in Section 4.3.3. Finally, we compared MMF's class-compositions to the hierarchical clusters obtained from agglomerative clustering of representations. The relationships in Figure 4(b-d) are not apparent in the corresponding dendrograms (see supplement, [32]) – for instance, the dependency of *chutney*/*salsa*/*salad* on several breads, or the disparity of *ketchup* from the others.

Overall, Figure 4(b–e) shows many of the summaries that a human may infer about the 12 classes in Figure 4(a). Apart from visualizing deep representations, such MMF



(a) **12 classes** (b) **5^{th} order (FC7 layer reps.)** (c) **5^{th} order (FC7 layer reps.)**

(d) **4^{th} order (FC7 layer reps.)** (e) **5^{th} order (conv3 layer reps.)** (f) **5^{th} order (Pixel reps.)**

Figure 4. **Hierarchy and Compositions of VGG-S [9] representations inferred by MMF.** (a) The **12** classes, (b,c) Hierarchical structure from $\mathbf{5^{th}}$ order MMF, (d) the structure from a $\mathbf{4^{th}}$ order MMF, and (e,f) compositions from $\mathbf{3^{rd}}$ conv. layer (VGG-S) and inputs. $\tilde{\boldsymbol{m}} = \mathbf{0.1}\boldsymbol{m}$ (from Algorithm 3).

graphs are vital exploratory tools for category/scene understanding from unlabeled representations in transfer and multi-domain learning [2]. This is because, by comparing the MMF graph *prior* to inserting the new unlabeled instance to the one *after* insertion, one can infer whether the new instance contains non-trivial information that cannot be expressed as a composition of existing categories.

4.3.3 The flow of MMF graphs: An exploratory tool

Figure 4(f) shows the compositions from the 5^{th} order MMF on the *input* (pixel-level) data. These features are non-informative, and clearly, the classes whose RGB values correlate are at $l = 0$ in Figure 4(f). But most importantly, comparing Figure 4(b,e) we see that $l = 1$ and 2 have the same compositions. One can construct visualizations like Figure 4(b,e,f) for all the layers of the network. Using this trajectory of the class compositions, one can ask whether a new layer needs to be added to the network (a vital aspect for model selection in deep networks [19]). This is driven by the saturation of the compositions – if the last few levels' hierarchies are similar, then the network has already learned the information in the data. On the other hand, variance in the last levels of MMFs implies that adding another network layer may be beneficial. The saturation at $l = 1, 2$ in Figure 4(b,e) (see supplement for remaining layers' MMFs) is one such example. If these 8 classes are a priority, then the predictions of the VGG-S' 3^{rd} convolutional layer may already be *good enough*. Such constructs can be tested across other layers and architectures (see supplement for MMFs from AlexNet, VGG-S and other networks).

5. Conclusions

We present an algorithm that uncovers multiscale structure of symmetric matrices by performing a matrix factorization. We showed that it is an efficient importance sampler for relative leveraging of features.We also showed how the factorization sheds light on the semantics of categorical relationships encoded in deep networks, and presented ideas to facilitate adapting/modifying their architectures.

Acknowledgments: The authors are supported by NIH AG021155, EB022883, AG040396, NSF CAREER 1252725, NSF AI117924 and 1320344/1320755.

References

[1] Inceptionism: Going deeper into neural networks. 2015. 6

[2] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 8

[3] C. Boutsidis, P. Drineas, and M. W. Mahoney. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems*, pages 153–161, 2009. 5, 6

[4] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013. 1

[5] E. J. Candes and D. L. Donoho. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Technical report, DTIC Document, 2000. 1

[6] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009. 1

[7] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014. 5

[8] S. Chandrasekaran, M. Gu, and W. Lyons. A fast adaptive solver for hierarchically semiseparable representations. *Calcolo*, 42(3-4):171–185, 2005. 1

[9] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014. 1, 6, 8

[10] A. M. Cheriyadat and R. J. Radke. Non-negative matrix factorization of partial track data for motion segmentation. In *2009 IEEE 12th International Conference on Computer Vision*, pages 865–872. IEEE, 2009. 1

[11] R. R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006. 1

[12] P. F. de Aguiar, B. Bourguignon, M. Khots, D. Massart, and R. Phan-Than-Luu. D-optimal designs. *Chemometrics and Intelligent Laboratory Systems*, 30(2):199–210, 1995. 6

[13] F. De la Torre and M. J. Black. Robust principal component analysis for computer vision. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 362–369. IEEE, 2001. 1

[14] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. *arXiv preprint arXiv:1506.02753*, 2015. 6

[15] K. J. Friston, A. P. Holmes, K. J. Worsley, J.-P. Poline, C. D. Frith, and R. S. Frackowiak. Statistical parametric maps in functional imaging: a general linear approach. *Human brain mapping*, 2(4):189–210, 1994. 5

[16] M. Gavish and R. R. Coifman. Sampling, denoising and compression of matrices by coherent matrix organization. *Applied and Computational Harmonic Analysis*, 33(3):354–369, 2012. 1

[17] W. Hwa Kim, H. J. Kim, N. Adluru, and V. Singh. Latent variable graphical model selection using harmonic analysis: applications to the human connectome project (hcp). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2443–2451, 2016. 1

[18] A. Hyvärinen. Independent component analysis: recent advances. *Phil. Trans. R. Soc. A*, 371(1984):20110534, 2013. 1

[19] V. K. Ithapu, S. N. Ravi, and V. Singh. On architectural choices in deep learning: From network structure to gradient convergence and parameter estimation. *arXiv preprint arXiv:1702.08670*, 2017. 8

[20] V. K. Ithapu, V. Singh, O. C. Okonkwo, et al. Imaging-based enrichment criteria using deep learning algorithms for efficient clinical trials in mild cognitive impairment. *Alzheimer's & Dementia*, 11(12):1489–1499, 2015. 6

[21] R. Kondor, N. Teneva, and V. Garg. Multiresolution matrix factorization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1620–1628, 2014. 2, 3

[22] R. Kondor, N. Teneva, and P. K. Mudrakarta. Parallel mmf: a multiresolution approach to matrix computation. *arXiv preprint arXiv:1507.04396*, 2015. 2, 5

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[24] G. Kutyniok et al. *Shearlets: Multiscale analysis for multivariate data*. Springer Science & Business Media, 2012. 1

[25] A. B. Lee, B. Nadler, and L. Wasserman. Treelets: an adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, pages 435–471, 2008. 1

[26] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647. ACM, 2005. 6

[27] P. Ma, M. W. Mahoney, and B. Yu. A statistical perspective on algorithmic leveraging. *Journal of Machine Learning Research*, 16:861–911, 2015. 5

[28] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989. 2

[29] B. S. Manjunath and W.-Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence*, 18(8):837–842, 1996. 1

[30] Y. Meyer. Wavelets-algorithms and applications. *Wavelets-Algorithms and applications Society for Industrial and Applied Mathematics Translation., 142 p.*, 1, 1993. 1

[31] F. Mezzadri. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*, 2006. 5

[32] J. C. Peterson, J. T. Abbott, and T. L. Griffiths. Adapting deep network features to capture psychological representations. *arXiv preprint arXiv:1608.02164*, 2016. 6, 7

[33] S. S. Rautaray and A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015. 5

[34] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010. 1

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1

[36] B. Savas, I. S. Dhillon, et al. Clustered low rank approximation of graphs in information science applications. In *SDM*, pages 164–175. SIAM, 2011. 1

[37] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 6

[38] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European conference on computer vision*, pages 709–720. Springer, 1996. 1

[39] N. Teneva, P. K. Mudrakarta, and R. Kondor. Multiresolution matrix compression. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1441–1449, 2016. 2, 3

[40] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991. 1

[41] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2013. 6

[42] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009. 1

[43] J. Xu, V. K. Ithapu, L. Mukherjee, J. M. Rehg, and V. Singh. Gosus: Grassmannian online subspace updates with structured-sparsity. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3376–3383, 2013. 1

[44] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. 6

[45] T. Zhang, B. Ghanem, and N. Ahuja. Robust multi-object tracking via cross-domain contextual information for sports video analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 985–988. IEEE, 2012. 6

[46] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006. 2