

---

# Tensor-Dictionary Learning with Deep Kruskal-Factor Analysis

---

Andrew Stevens  
PNNL & Duke U.

Yunchen Pu  
Duke Univ.

Yannan Sun  
PNNL

Gregory Spell  
Duke Univ.

Lawrence Carin  
Duke Univ.

## Abstract

A multi-way factor analysis model is introduced for tensor-variate data of any order. Each data item is represented as a (sparse) sum of Kruskal decompositions, a Kruskal-factor analysis (KFA). KFA is nonparametric and can infer both the tensor-rank of each dictionary atom and the number of dictionary atoms. The model is adapted for online learning, which allows dictionary learning on large data sets. After KFA is introduced, the model is extended to a deep convolutional tensor-factor analysis, supervised by a Bayesian SVM. The experiments section demonstrates the improvement of KFA over vectorized approaches (*e.g.*, BPFA), tensor decompositions, and convolutional neural networks (CNN) in multi-way denoising, blind inpainting, and image classification. The improvement in PSNR for the inpainting results over other methods exceeds 1dB in several cases and we achieve state of the art results on Caltech101 image classification.

## 1 Introduction

The first data analysis applications of tensors saw the development of the Tucker and canonical decomposition (CANDECOMP) in chemometrics and the parallel factor model (PARAFAC) in linguistics; the CANDECOMP/PARAFAC is also called the Kruskal decomposition (KD). Most recently, tensor methods have been applied in signal/image processing and machine learning (Cichocki et al., 2015).

Tensor decompositions are limited in their representation size, and the number of atoms/factor-loadings can be at most the rank of the tensor (*i.e.*, they are

not overcomplete). Following on the success of the K-SVD dictionary learning algorithm (Aharon et al., 2006), several tensor-dictionary learning approaches have been proposed, *e.g.* (Hawe et al., 2013). The goal of overcomplete dictionary learning is to find a set of atoms that can represent each data item as a weighted sum of a small subset of those atoms. A dictionary is called overcomplete when the number of atoms  $K$  is greater than the dimension  $P$  of each data item, this is closely related to frame theory (Christensen, 2008). Overcompleteness is desirable because it offers robustness to noise, increased sparsity, and improved interpretability (Lewicki and Sejnowski, 2000). An overview of non-tensor dictionary learning approaches can be found in (Mairal et al., 2014).

Previous work in overcomplete tensor-dictionary learning has focused on separable models (Sironi et al., 2015; Hawe et al., 2013; Gribonval et al., 2015). Imposing separable structure means that each atom is a rank-1 tensor. Moreover, they consider only “shallow” representations and are limited to 3-way data. In this paper we develop Bayesian nonparametric models for tensor-variate data with tensor-variate dictionary atoms and learn the rank of each atom.

The concept of overcompleteness becomes unclear when we move to high-order data. Vectors can be represented as a sum of basis elements, so an overcomplete dictionary gives more elements than needed to represent a vector. In contrast, Matrices (2-way tensors) can be represented as a sum of rank-1 matrices (*i.e.*, SVD). If a particular matrix is  $m_1 \times m_2$ , an overcomplete vectorized representation will have more than  $m_1 m_2$  atoms, but a “rank-overcomplete” representation only needs more than  $\min\{m_1, m_2\}$  atoms.

To address the limitations of current (tensor) dictionary learning approaches, we propose a nonparametric tensor-factor analysis model. The proposed Kruskal-factor analysis (KFA) is compatible with tensors of any order and capable of performing blind inpainting. Moreover, we provide extensions to deep convolutional learning and implement a minibatch learning approach. Finally, experimental results on image processing and classification tasks are shown with state of the art re-

sults for 2D & 3D inpainting and Caltech 101, and evidence that rank is an important aspect of overcomplete modeling.

## 2 Preliminaries

**Dictionary learning:** Formally, dictionary learning is a factorization of a data tensor  $\mathcal{X}$ . If the data are column vectors we have  $\mathbf{X} = \mathbf{D}\mathbf{W} + \mathbf{E}$ , where the  $N$  data vectors are combined into the matrix  $\mathbf{X} \in \mathbb{R}^{P \times N}$ , the dictionary  $\mathbf{D} \in \mathbb{R}^{P \times K}$ , the weight matrix  $\mathbf{W} \in \mathbb{R}^{K \times N}$ , and  $\mathbf{E}$  is the residual/noise. When  $K > P$  it is necessary to impose that  $\mathbf{W}$  is sparse to ensure identifiability.

Beta-process factor analysis (BPFA) is a Bayesian nonparametric model for dictionary learning (Paisley and Carin, 2009). In BPFA we assume that the noise is Gaussian and impose sparsity using a truncated beta-Bernoulli process. The BPFA model is specified as

$$\begin{aligned} \mathbf{x}_n &\sim \mathcal{N}(\mathbf{D}\mathbf{w}_n, \gamma_\epsilon^{-1} \mathbf{I}_P), & \mathbf{w}_n &= \mathbf{s}_n \circ \mathbf{z}_n \\ \mathbf{d}_k &\sim \mathcal{N}(0, P^{-1} \mathbf{I}_P), & \mathbf{s}_n &\sim \mathcal{N}(0, \gamma_s^{-1} \mathbf{I}_K) \\ z_{kn} &\sim \text{Bernoulli}(\pi_k), & \pi_k &\sim \text{Beta}(a_\pi/K, b_\pi(K-1)/K) \end{aligned} \quad (1)$$

where  $\gamma_\epsilon, \gamma_s$  have noninformative gamma priors with all hyperparameters set to  $10^{-6}$ , the prior for  $\pi_k$  is nearly uniform with  $a_\pi = K$ ,  $b_\pi = 1$ , and  $\circ$  is the Hadamard elementwise product. The probabilities  $\pi_k$  determine how likely an atom is to be used and their hyperparameters directly affect sparsity. Other priors for the weights have been proposed in (Wang and Carin, 2012; Yuan et al., 2014; Zhang et al., 2011, 2016). Gaussian process atoms were used in (Xing et al., 2012).

**Tensor decomposition:** A tensor is a mathematical object describing linear transformation rules, and once we fix a basis we can represent a tensor as a multidimensional array of numbers. In data analysis, it is common to call a multidimensional array a tensor. An entry of a  $T$ -way or order- $T$  tensor  $\mathcal{X} \in \mathbb{R}^{m_1 \times \dots \times m_T}$  is indicated by a position vector  $\mathbf{i} = [i_1, \dots, i_T]$ , where  $i_t$  is an integer on  $[1, m_t]$ ; we denote the  $\mathbf{i}^{\text{th}}$  entry of the tensor  $\mathcal{X}$  as  $x_{\mathbf{i}}$ . We use  $\otimes$  to represent the Kronecker product. We also use the mapping  $\mathbf{x} = \text{vec}(\mathcal{X})$ , defined by  $x_i = \text{vec}(\mathcal{X})_{i_1+i_2m_1+\dots+i_Tm_1 \dots m_{T-1}}$ .

If we consider a matrix  $\mathbf{X} \in \mathbb{R}^{m_1 \times m_2}$  we can write it as  $\mathbf{X} = \sum_{r=1}^{m_1 \wedge m_2} \lambda_r \mathbf{u}_r^{(1)} \mathbf{u}_r^{(2)\top}$ , where  $\lambda_r \in \mathbb{R}$ ,  $\mathbf{u}_r^{(1)} \in \mathbb{R}^{m_1}$ ,  $\mathbf{u}_r^{(2)} \in \mathbb{R}^{m_2}$ . This is the SVD. A generalization of the SVD to a  $T$ -way tensor  $\mathcal{X}$  is called the canonical polyadic decomposition (PD). The canonical PD is written as  $\mathcal{X} = \sum_{r=1}^{R_M} \lambda_r \otimes_{t=1}^T \mathbf{u}_r^{(t)}$ , where  $R_M$  is the maximum rank of tensors in  $\mathbb{R}^{m_1 \times \dots \times m_T}$ ,  $\mathbf{u}_r^{(t)} \in \mathbb{R}^{m_t}$ . The rank of a tensor is the number of nonzero  $\lambda_r$ .

Recently, the multiplicative gamma process CANDECOMP/PARAFAC (MGP-CP) (Rai et al., 2014), a

Bayesian nonparametric KD, was proposed. MGP-CP can additionally infer the rank of a tensor. In MGP-CP, the MGP (Bhattacharya and Dunson, 2011) imposes that the singular values should shrink in absolute value as  $r$  increases. The MGP shrinks the variance of a zero mean Gaussian (the prior for  $\lambda_r$ ), so that for large  $r$  the singular value will be near zero with high probability. When a singular value shrinks enough (*e.g.*,  $\lambda_r < 10^{-6}$ ) we assume it is zero. The numerical rank is given by the number of nonzero  $\lambda_r$  after thresholding. The MGP-CP is described entrywise,  $x_{\mathbf{i}} = \sum_{r=1}^R \lambda_r \prod_{t=1}^T u_{i_t r}^{(t)} + \epsilon_{\mathbf{i}}$  is the  $\mathbf{i}^{\text{th}}$  entry of  $\mathcal{X}$ ,  $R$  is the maximum desired rank and  $\epsilon_{\mathbf{i}}$  is Gaussian noise. The MGP-CP is specified as

$$\begin{aligned} x_{\mathbf{i}} &\sim \mathcal{N}\left(\sum_{r=1}^R \lambda_r \prod_{t=1}^T u_{i_t r}^{(t)}, \tau_\epsilon^{-1}\right), & \mathbf{u}_r^{(t)} &\sim \mathcal{N}(0, \omega_{rt}^{-1} \mathbf{I}_{m_t}) \\ \lambda_r &\sim \mathcal{N}(0, \tau_{kr}^{-1}), & \tau_{kr} &= \prod_{i=1}^r \delta_i, & \delta_i &\sim \text{Gamma}(\alpha, 1) \end{aligned} \quad (2)$$

and  $\tau_\epsilon, \omega_{rt}$  have noninformative gamma priors with all hyperparameters set to  $10^{-6}$ . The shape parameter  $\alpha$  for the MGP is usually set to 2 or 3. If  $\alpha > 1$ , the error in  $\sum_{r=1}^R \lambda_r \prod_{t=1}^T u_{i_t r}^{(t)}$  converges to zero exponentially fast as  $R \rightarrow \infty$  (Rai et al., 2014).

**Tensor Rank:** The maximum rank  $R_M$  is usually not  $\min_t m_t$  like the matrix case. Finding the canonical PD of a tensor is a difficult computational problem. It is possible to find PD with a preset  $R$  (that could be different from the actual rank), these are called KD.

A useful characterization of tensors is balance, which we use to set  $R$  for KFA. Assuming  $m_T$  is the largest dimension, a tensor is balanced when  $m_T \leq \prod_{t=1}^{T-1} m_t - \sum_{t=1}^{T-1} (m_t - 1)$  and unbalanced when  $m_T \geq 1 + \prod_{t=1}^{T-1} m_t - \sum_{t=1}^{T-1} (m_t - 1)$ . An unbalanced tensor has generic rank  $R_G = \min\{m_T, \prod_{t=1}^{T-1} m_t\} \leq R_M$ , similar to matrices. Balanced tensors are expected to have generic rank  $R_E = \lceil \prod_t m_t / (\sum_t m_t - T + 1) \rceil \leq R_M$  (Abo et al., 2009).

With the notions of tensor rank introduced we can define rank-overcomplete. A tensor factorization is rank-overcomplete when the number of (tensor) atoms is greater than the rank of the tensor. We note that this definition also applies to slices of tensors, *e.g.* data items that have been stacked into a larger tensor. For example, in 2D  $\mathbf{X} = \sum_{k=1}^K \lambda_k \mathbf{u}_k^{(1)} \mathbf{u}_k^{(2)\top}$ , is an overcomplete decomposition if  $K > \min\{m_1, m_2\}$ .

## 3 Kruskal Factor Analysis

The most natural generalization of dictionary learning to tensors is to model the data as the product of a dictionary tensor and a weight matrix along the  $(T+1)^{\text{st}}$  dimension of the dictionary. We define the mode- $t$  tensor product  $\times_t$  with an example: the mode-3 product of a tensor  $\mathcal{D} \in \mathbb{R}^{m_1 \times m_2 \times K}$  and a matrix  $\mathbf{W} \in \mathbb{R}^{K \times N}$  yields  $\mathcal{X} = \mathcal{D} \times_3 \mathbf{W} \in \mathbb{R}^{m_1 \times m_2 \times N}$  with

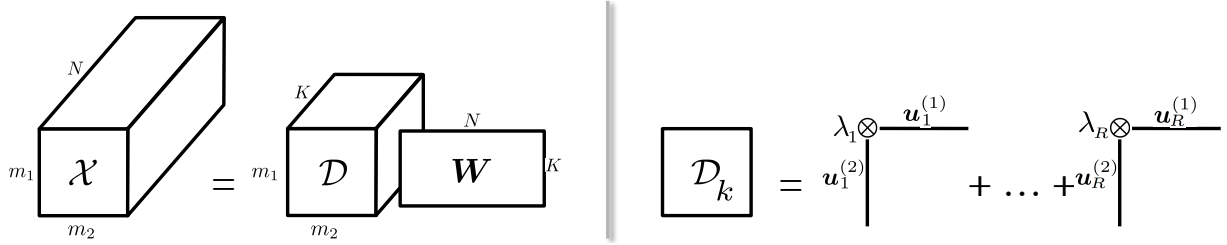


Figure 1: An illustration of the KFA Model for 2-way data. The left equation shows the mode-3 tensor product generalization of dictionary learning. The right equation shows the representation of dictionary atom  $k$  using the KD, for 2-way data this is the SVD. The illustration clearly shows the reduction in the number of parameters compared to a non-decomposed dictionary. Each atom in a non-decomposed dictionary will have  $\prod m_t$  parameters, while our approach has  $R(1 + \sum m_t)$ . The ability to specify  $R$  gives our model greater flexibility. Moreover, by applying shrinkage on  $\lambda_r$  and learning the shrinkage rate  $\alpha$ , our model automatically adjusts model complexity during learning.

$x_{i_1 i_2 i_3} = \sum_{k=1}^K d_{i_1 i_2 k} w_{k i_3}$ . The factorization of  $\mathcal{X}$  is illustrated for 2-way data in Figure 1.

We extend dictionary learning to arbitrary order tensors by inferring the ‘‘singular’’ vectors/values of each atom via KD. The Kruskal representation for each atom is used in the likelihood. It is important to note that when we learn the model we are not replacing the dictionary with a low-rank approximation. The singular vectors/values are drawn from their posteriors, thus they are directly affected by the data; if the data requires full-rank atoms they will be learned. We call this new model Kruskal-factor analysis (KFA). The model is specified as follows:

$$\begin{aligned} \mathcal{X}_n &\sim \mathcal{N}(\text{vec}[\mathcal{D}_{T+1} \times (\mathbf{s}_n \circ \mathbf{z}_n)], \gamma_\epsilon^{-1} \mathbf{I}), d_{ik} = \sum_{r=1}^R \lambda_{rk} \prod_{t=1}^T u_{i_t r}^{(kt)} \\ u_{i_t r}^{(kt)} &\sim \mathcal{N}(0, m_t^{-1}), \lambda_{kr} \sim \mathcal{N}(0, \tau_{kr}^{-1}), \tau_{kr} = \prod_{i=1}^r \delta_{ki} \\ s_{kn} &\sim \mathcal{N}(0, \gamma_s^{-1}), z_{kn} \sim \text{Bernoulli}(\pi_k) \\ \pi_k &\sim \text{Beta}(a_\pi/K, b_\pi(K-1)/K), \delta_{ki} \sim \text{Gam}(\alpha, 1) \end{aligned} \quad (3)$$

where  $\mathcal{X}_n \in \mathbb{R}^{m_1 \times \dots \times m_T}$  is the  $n^{\text{th}}$  data item and  $d_{ik}$  is the  $i = [i_1, \dots, i_T]$  element of the  $k^{\text{th}}$  atom. The hyperparameters for  $\gamma_\epsilon, \gamma_s$  are set to  $10^{-6}$  to give a noninformative prior and  $a_\pi = K, b_\pi = 1$ . The shape parameter  $\alpha$  for singular value shrinkage is inferred, this is discussed below. Note that the negative log-likelihood is simply  $\|\mathcal{X} - \mathcal{D}_{T+1} \times (\mathbf{S} \circ \mathbf{Z})\|_F^2$ , the atom decompositions impose tensor structure and the priors correspond to regularizers. By design, our model is fully locally conjugate and can be implemented efficiently as a Gibbs sampler. A Gibbs sampler produces a Markov chain whose stationary distribution is the joint distribution of the model, and the mode of a set of samples (from the chain) is a maximum a posteriori (MAP) solution to the corresponding regularized optimization problem.

In order to gain intuition about the model, consider the case when we fix  $R = 1$ . Every atom  $\mathcal{D}_k$  will be rank 1, so each data item is a sum of rank 1 tensors. This is a KD where the rank 1 tensors are selected from a dictionary. The rank of the output tensor is

determined by the number of nonzero entries in  $\mathbf{z}_n$  ( $\min\{\text{nnz}(\mathbf{z}_n), R_M\}$ ), with  $\mathbf{s}_n$  acting as singular values. Increasing  $R$  yields:  $\mathcal{X}_n = \sum_{k,r} s_{nk} z_{nk} \lambda_{rk} \otimes_t \mathbf{u}_r^{(kt)}$ , which is a KD using rescaled singular values and  $s_{nk} z_{nk}$ .

**Dictionary updates:** With a careful derivation we have found an update procedure that seamlessly converts BPFA into a tensor model. The naive approach, specifying the likelihood elementwise, as in MGP-CP, is poorly suited to computation, since the singular values/vectors must be updated for every entry of the data tensor—making the dictionary update complexity  $O(NKT^2 R(1 + \sum m_t) \prod m_t)$  for a single Gibbs iteration (without including the expensive residual computation). In our implementation, entire atoms are updated at once. Moreover, the atom-wise updates clearly show the overhead of the KD. The first step in the update is to compute the mean and covariance just as in BPFA, these are then used to sample all of the atom parameters. With the following reparameterizations

$$\begin{aligned} d_{ik} &= \lambda_{rk} \prod_{t' \neq t} u_{i_{t'} r}^{(kt')} u_{i_t r}^{(kt)} + \sum_{r' \neq r} \lambda_{r'k} \prod_{t=1}^T u_{i_t r'}^{(kt)} = a_{i_t} u_{i_t r}^{(kt)} + b_{i_t} \\ &= \prod_{t=1}^T u_{i_t r}^{(kt)} \lambda_{rk} + \sum_{r' \neq r} \lambda_{r'k} \prod_{t=1}^T u_{i_t r'}^{(kt)} = f_i \lambda_{rk} + g_i, \end{aligned} \quad (4)$$

we derive closed form updates for dictionary atoms:

$$\begin{aligned} u_{i_t r}^{(kt)} &\sim \mathcal{N}(\hat{\mu}, \hat{\omega}^{-1}) & \lambda_{kr} &\sim \mathcal{N}(\hat{\nu}, \hat{\tau}^{-1}) \\ \hat{\omega} &= m_t + \sum_{i: i_t=m} \gamma_{i_t k}^{\mathcal{D}_k} (a_{i_t})^2 & \hat{\tau} &= \tau_{kr} + \gamma_\epsilon \sum_{n,i} (f_i s_{kn})^2 \\ \hat{\mu} &= \frac{1}{\hat{\omega}} \sum_{i: i_t=m} (\mu_{i_t k}^{\mathcal{D}_k} - b_{i_t}) a_{i_t} & \hat{\nu} &= \frac{\gamma_\epsilon}{\hat{\tau}} \sum_{n,i} (h_{in} - g_i s_{kn}) f_i s_{kn}, \end{aligned} \quad (5)$$

where  $\gamma_{i_t k}^{\mathcal{D}_k}, \mu_{i_t k}^{\mathcal{D}_k}$  are the BPFA updates for the precision and mean (without multiplying the covariance) and  $\mathcal{H} = \mathcal{X} - \mathcal{D}_{T+1} \times (\mathbf{S} \circ \mathbf{Z})$  is the current residual.

The complexity for sampling  $\mathcal{D}$  in BPFA is  $O(KM^2 \sum n_k^2)$  (with inpainting (Zhou et al., 2012)), where  $M = \prod m_t$  and  $n_k$  is the number of data items using atom  $k$ . This arises from the residual updates (which includes an elementwise matrix product and

an outer product of matrices), which dominate. For a  $T$ -way tensor KFA adds  $O(KT^2R(1 + \sum m_t))$  operations, so the complexity for sampling  $\mathcal{D}_k$  in KFA is the same as BPFA (since  $M^2 \sum n_k^2 \gg T^2R(1 + \sum m_t)$ , for practical  $N$ ). The weight matrices  $\mathbf{S}$  and  $\mathbf{Z}$  have the same updates in BPFA and KFA, so the complexity of one Gibbs iteration of KFA is the same as BPFA.

**Learning the shape parameter  $\alpha$ :** The shape parameter is critical to model quality because it determines the shrinkage rate for the singular values. We found that fixing  $\alpha$  produces suboptimal results. Updating  $\alpha$  while the model is learned is important because it adjusts the complexity of the dictionary. Note that a similar approach was taken in (Rai et al., 2014), where the rank of the tensor was increased or decreased *ad hoc* depending on whether the singular values are larger than or smaller than preset thresholds.

KFA specifies a prior for  $\alpha$ , which connects the dictionary atoms. The conjugate prior for a gamma distribution with known rate  $\beta = 1$  is given by  $p(\alpha; a, b) \propto \frac{a^{\alpha-1}}{\Gamma(\alpha)^b}$ , so the posterior distribution is

$$p(\alpha|\{\delta_{ki}\}, a, b) = \text{Gamma}(\delta_{ki}; \alpha, 1)p(\alpha; a, b) \propto (a \prod_{k,i} \delta_{ki})^{\alpha-1} \Gamma(\alpha)^{-b-RK}. \quad (6)$$

Although this prior gives straightforward updates, it is not a standard distribution to sample from. We use numerical integration to perform inverse-CDF sampling. Using the conditional MAP solution for  $\alpha$  also works well and is simpler. The MAP solution is closed form:  $\psi^{-1}\left(\frac{\log(a) + \sum_{k,i} \delta_{ki}}{b + RK}\right)$ , where  $\psi^{-1}$  is the inverse digamma function.

We choose the hyperparameters  $a = 10^6$ ,  $b = 10^{-6}$  and set the initial value for  $\alpha = 10^6$ . This causes the model to gradually increase from rank 1 atoms up to the inferred minimal rank. Without sampling  $\alpha$ , the parameters  $R$  and  $\alpha$  must be selected manually. In our approach we set  $R = R_E - 1$  or  $R = R_G - 1$ , depending on whether each data item is balanced or not, and let the model learn  $\alpha$ . We reduce the rank by 1 to prevent overfitting.

**Scalability:** We use the Bayesian conditional density filter (BCDF) (Guhaniyogi et al., 2014) to allow our model to scale to large datasets. Essentially, BCDF allows MCMC models to be trained online or by iterating over subsets of training data. This application of BCDF is not the same as the original authors intended. In their work, only one epoch was used. Our BCDF approach removes the need to load an entire dataset into memory. The details are given in the supplement.

## 4 Deep Convolutional KFA

Instead of matrix-vector multiplication between the atoms and the sparse weights we can use convolution. This will imbue the model with translational invariance; specializing BPFA with a convolutional likelihood is called convolutional factor analysis (CFA) (Chen et al., 2011). In the setting of images we use the 2D (spatial) convolution, and for multi-channel images the convolution is applied separately to each channel. The likelihood for CFA is  $\mathbf{X}_n \sim \mathcal{N}(\sum_k \mathbf{D}_k * \mathbf{W}_{kn}, \gamma_\epsilon^{-1} \mathbf{I}_P)$ , which can be converted into a tensor model by applying KD to  $\mathbf{D}_k$  and applying the same priors as KFA. Note that the weights  $\mathbf{W}_{kn}$  are matrices and the data  $\mathbf{X}_n \in \mathbb{R}^{M_x \times M_y \times M_c}$  and dictionary atoms  $\mathbf{D}_k \in \mathbb{R}^{m_x \times m_y \times M_c}$  are tensors (*e.g.*,  $M_c = 3$  for RGB images). Usually, the dimensions of the dictionary atoms are much smaller than the image dimensions ( $m_x \ll M_x$  and  $m_y \ll M_y$ ). The spatially-dependent activation weights for dictionary atom  $k$ , image  $n$ , are  $\mathbf{W}_{kn} \in \mathbb{R}^{(M_x - m_x + 1) \times (M_y - m_y + 1)}$ . More details can be found in Chen et al. (2011); Pu et al. (2014, 2016b). The dictionary atoms in CFA, although they are tensors, are sampled from a multivariate Gaussian.

In a deep architecture, the set of weights  $\{\mathbf{W}_{kn}\}_{k=1}^K$  for image  $n$  are represented in terms of (different) convolutional dictionary atoms. When an  $L$ -layer deep model is built, the input of the  $\ell^{\text{th}}$  layer is usually composed of a pooled version of the output of the layer below, the  $(\ell - 1)^{\text{st}}$  layer. We can formulate this deep deconvolutional model via these two contiguous layers:

$$\mathbf{X}_n^{(\ell-1)} = \sum_{k=1}^{K_{\ell-1}} \mathbf{D}_k^{(\ell-1)} * \mathbf{W}_{kn}^{(\ell-1)}, \quad \mathbf{X}_n^{(\ell)} = \sum_{k=1}^{K_\ell} \mathbf{D}_k^{(\ell)} * \mathbf{W}_{kn}^{(\ell)}, \quad (7)$$

$$\mathbf{W}_{kn}^{(\ell-1)} := \text{unpool}(\mathbf{X}_{kn}^{(\ell)}). \quad (8)$$

The weight matrices  $\mathbf{W}_{kn}^{(\ell)}$  in the  $\ell^{\text{th}}$ -layer become the inputs to the  $(\ell + 1)^{\text{st}}$ -layer after pooling. The input tensor  $\mathbf{X}_n^{(\ell+1)}$  is constituted by stacking the  $K_\ell$  spatially aligned  $\mathbf{X}_{kn}^{(\ell+1)}$ . The  $(\ell + 1)^{\text{st}}$ -layer inputs are tensors with the third-dimension of size  $K_\ell$ , the dictionary size in the  $\ell^{\text{th}}$ -layer. This is deep CFA. When a stochastic unpooling process is employed in (8), and appropriate priors are imposed on dictionary and feature parameters, the model developed in (7) constitutes a generative model for images called the deep generative deconvolutional network (DGDN) (Pu et al., 2016b).

Within each layer of the deep CFA and DGDN model we employ KFA, resulting in deep CKFA and Kruskal DGDN. A major difference from multiplicative KFA is that both the weights and the dictionary are tensors. Kruskal DGDN can be supervised by connecting a Bayesian SVM (Polson and Scott, 2011) to the top layer weights  $\mathbf{W}_{kn}^L$  (Pu et al., 2016b).

## 5 Related Work

KFA is different from K-SVD and BPFA, since their atoms are vectors, and thus atom-rank is not defined. Moreover, in the experiments we found that BPFA prefers full-rank atoms (determined by first reshaping), while KFA prefers reduced-rank atoms. In addition, the posterior distribution for a vectorized atom in KFA will have a Kronecker structured covariance (with off-diagonal interactions), whereas BPFA atoms have a diagonal posterior covariance.

The main difference of KFA and any Tucker-like decomposition, *e.g.* sparse Bayesian Tucker decomposition (sBTD) (Zhao et al., 2015), TenSR (Qi et al., 2016), Tensor Analyzers (Tang et al., 2013), or MGP-CP, is that KFA is rank-overcomplete, but Tucker-like decompositions can have at most  $R_M$  (a similar condition holds for multilinear-rank). Another difference from decompositions is that KFA infers several tensors simultaneously (the atoms). Further, we learn the rank and sparsity simultaneously. We also extend MGP-CP by learning the shrinkage rate. Without inferring  $\alpha$ , KFA performance suffers. MGP-CP is very sensitive to this parameter—techniques are proposed in (Rai et al., 2014) for tuning  $\alpha$ . Moreover, the rank being inferred is different (atoms vs. data tensor), and the atoms could be balanced or unbalanced, but for  $N \geq M$  the data tensor is usually unbalanced.

Some recent work has been done using a separable ( $R = 1$ ) tensor structure for the dictionary. One approach, separable dictionary learning (Hawe et al., 2013), considered two-way data (grayscale image patches) and claimed generalizing to higher order data was straightforward, but did not show any results in this direction. Their optimization algorithm is significantly more complicated than the separable convolution approach in (Sironi et al., 2015) and for grayscale denoising the performance is not as good as BPFA. The separable multiplicative structure is also considered in (Gribonval et al., 2015), where the sample complexity for various dictionary learning approaches is considered.

A separate line of work is concerned with high-order cumulant decomposition, which is used in, *e.g.* independent component analysis (Comon, 1994). Cumulants are symmetric, so all of the factor matrices are identical (Comon et al., 2008). This is very different from direct analysis of tensor data.

## 6 Experiments

In this section we show that maintaining the tensor structure of the data is beneficial in several tasks. We compare against vectorized factor analysis to show that keeping structure can boost performance. All

experiments use the default parameters discussed above and none of the parameters were tuned or optimized. We implemented our models in Matlab. All latent parameters are initialized randomly, except  $\alpha$  which is set to  $10^6$  to initialize an essentially separable model. In the tables, bold represents the best result.

### 6.1 Image denoising and inpainting

Image denoising is a standard image processing task, and BPFA has been shown to produce very good denoising results without *a priori* knowledge of the noise level. Inpainting is another image processing task, and is a special case of compressive sensing. The goal in the inpainting task is to impute missing pixels. Simultaneous inpainting and denoising has become an important step in scientific imaging (Stevens et al., 2014; Kovarik et al., 2016). The model details for inpainting and compressive sensing are given in (Zhou et al., 2012). Finally, while we do not give direct comparisons to K-SVD Aharon et al. (2006), BPFA has been shown to be, in general, superior to K-SVD for denoising.

All of the image processing methods are patch-based. The dataset  $\mathcal{X} \in \mathbb{R}^{B \times B \times C \times N}$  is extracted from the image on a regular grid where the corner of each patch is separated from the other patches by  $(\Delta, \Delta)$  pixels ( $B$  is patch width/height,  $C$  is the number of colors). In these experiments 150 samples are obtained, with the first 40 used as burn-in. The remaining samples are reconstructed as an image, and then averaged to give the final reconstruction. This reconstruction approach is a type of Bayesian model averaging, we do not compute a MAP or mean-field solution. For the our stochastic learning approach, termed CDF-KFA, we use 10 epochs with 5 for burn-in, a batch size of 5000, and we update all of the local parameters one time after the first epoch. The time for a single epoch is about 3 times the iteration time for KFA on the full dataset (CDF-KFA has  $3 \times$  more updates). For inpainting, the sensing mask pixels are drawn *iid* from a Bernoulli distribution and the same mask is used for the different algorithms. We measure reconstruction error using peak signal to noise ratio (PSNR).

**Grayscale:** The grayscale experiments use a  $256 \times 256$  Barbara image. The image was chosen so that a direct comparison can be made with published BPFA results. The patch size is  $8 \times 8$  in these experiments.

We first examine the performance of KFA by varying the complexity (via dictionary size  $K$ ) and robustness to small sample sizes. The results of the complexity experiment are shown in Figure 2; the dictionary size ranges from 1 to 256,  $\Delta = 1$ , and for KFA  $R = 1$ . We see that KFA, with much fewer parameters, outperforms BPFA. The number of parameters for the

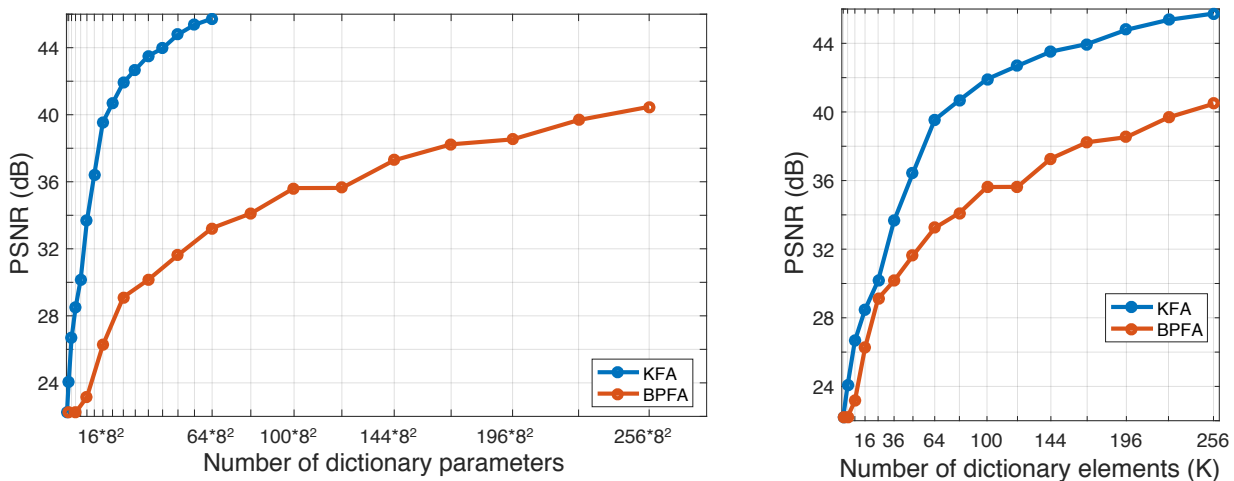


Figure 2: Reconstruction quality for varying model complexity. The same data is shown with different  $x$ -axis scalings: reconstruction PSNR as a function of the number of parameters (left) and as a function of the number of atoms (right).

KFA dictionary is  $KR(1 + \sum m_t) = 17K$  and BPFA has  $K \prod m_t = 64K$ . KFA also has a sizable jump in performance just above  $K = R_M = 8$ , but before  $K = \prod m_t$ , indicating that rank-overcompleteness is beneficial. For robustness to small sample sizes we train on datasets with  $\Delta = 1, 2, 4, 8$ , corresponding to 100%, 25%, 6.25%, and 1.56% of the patches; results are shown in Table 1 each entry is the mean of 10 runs; the standard deviation for KFA and BPFA was 0.1, for MGP-CP 1.2 and for CDF-KFA 0.6. We also compare to MGP-CP (using patches) with  $R = R_G = 64$  and our new update for  $\alpha$  using the same number of samples as KFA. KFA performs substantially better than BPFA and KFA is more consistent for smaller datasets. We also find that the MGP-CP, performs better than BPFA.

Table 1: Reconstruction PSNR (dB) for fractions of data.

$\Delta$	1	2	4	8
MGP-CP	47.23	45.68	44.82	42.76
BPFA	42.48	41.43	40.21	36.88
CDF-KFA	44.56	41.14	40.83	39.01
KFA	<b>50.36</b>	<b>49.71</b>	<b>47.84</b>	<b>44.26</b>

The results for simultaneous denoising and inpainting of the Barbara image are listed in Table 2 and example reconstructions for 20% inpainting are in the SM. We find that KFA generally outperforms BPFA. The BPFA results were reported in (Zhou et al., 2012). The dictionaries learned by each algorithm from the uncorrupted image are shown in Figure 3. Qualitatively, the KFA dictionary has a larger variety of structures. The rank (via SVD, with a threshold of  $10^{-6}$ ) of the KFA dictionary atoms is between 2 and 7 with 96% having rank 4 or 5. In contrast, every BPFA atom has full rank. KFA dictionaries for  $R = 1, 2, 4, 8$  learned from the uncorrupted image are shown in Figure 4. The dictionaries clearly show that KFA enforces different rank; in higher-rank dictionaries KFA learns both low-

and high-rank atoms.

The results for only inpainting are substantially better than BPFA. KFA gains more than 1 dB in 3 of the cases. For denoising cases KFA also produces superior results. In combined denoising and inpainting KFA performs better than BPFA in all but one case.

**Color:** In the color image experiment we use the castle image. The patch size is  $7 \times 7 \times 3$ ,  $\Delta = 1$ , and  $K = 512$ . Denoising and inpainting were performed separately and the reconstruction PSNRs for the castle image are shown in Table 3. The inpainting results improve upon BPFA (reported by (Zhou et al., 2012)), and the denoising results are comparable. We also obtain a PSNR of 50.94 dB on the uncorrupted image using non-overlapping patches. The dictionaries learned by BPFA and KFA are shown in Figure 3.

Table 2: Reconstruction PSNR (dB) for the Barbara image reduced pixel collection and additive noise. Top: BPFA, Bottom: KFA, Subtable: CDF-KFA.

$\sigma$	PIXELS				
	10%	20%	30%	50%	100%
0	23.47	26.87	29.83	35.60	42.94
	<b>24.24</b>	<b>27.92</b>	<b>30.90</b>	<b>36.03</b>	<b>50.43</b>
5	23.34	26.73	29.27	<b>33.61</b>	37.70
	<b>24.11</b>	<b>27.43</b>	<b>30.08</b>	33.51	<b>38.22</b>
10	23.16	26.07	28.17	31.17	34.31
	<b>23.51</b>	<b>27.18</b>	<b>28.54</b>	<b>31.24</b>	<b>34.44</b>
15	22.66	25.17	26.82	29.31	32.14
	<b>23.09</b>	<b>25.43</b>	<b>27.92</b>	<b>29.36</b>	<b>32.18</b>
20	22.17	24.27	25.62	27.90	30.55
	<b>22.47</b>	<b>24.57</b>	<b>26.09</b>	<b>28.72</b>	<b>30.71</b>
25	21.68	23.49	24.72	26.79	29.30
	<b>21.91</b>	<b>23.96</b>	<b>25.09</b>	<b>27.02</b>	<b>30.17</b>
PIXELS	10%	20%	30%	50%	100%
$\sigma$	0	5	10	15	20
	23.51	26.80	27.92	28.72	30.17

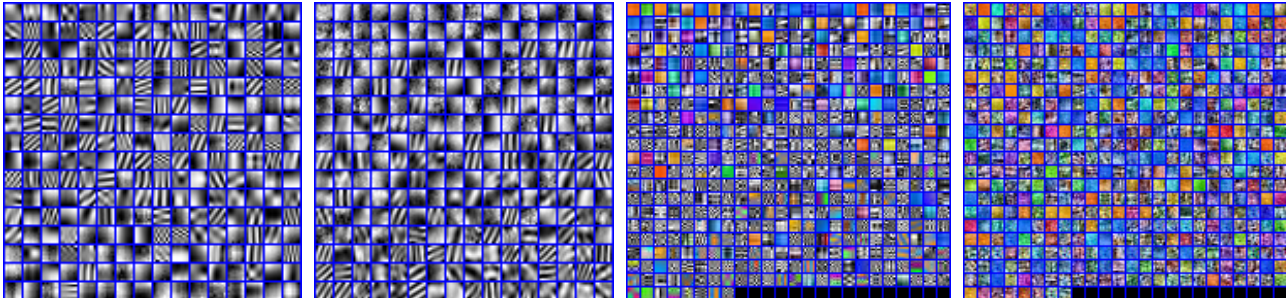


Figure 3: Dictionaries from clean images. Left to right: Barbara KFA, BPFA; Castle KFA, BPFA. Zoom for detail.

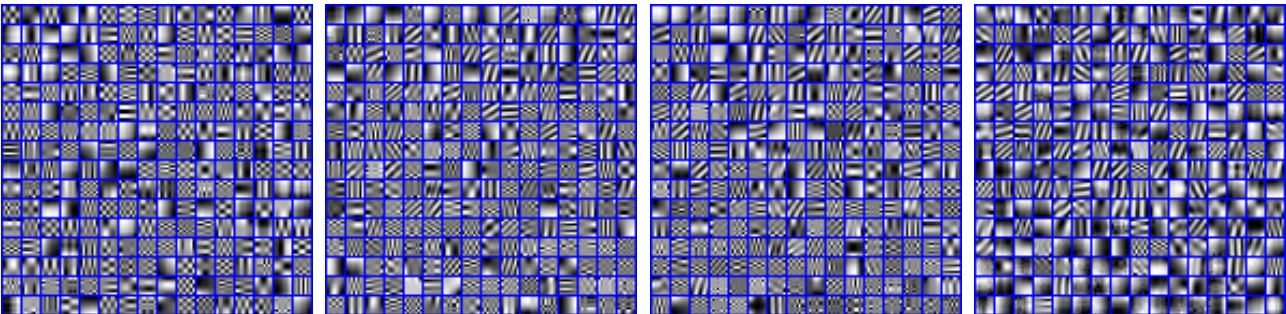
Figure 4: Barbara KFA dictionaries: from left to right  $R = 1, 2, 4, 8$ . For  $R = 1$  KFA learns a very DCT-like dictionary. As the rank increases diagonal structures are added. For  $R = 8$ , most atoms have rank less than 5, and none are full-rank.

Table 3: Reconstruction PSNR (dB) for the castle image.

PIXEL%	20	30	50	80
BPFA	29.12	32.02	36.45	41.51
CDF-KFA	28.95	31.87	37.18	44.60
KFA	<b>29.59</b>	<b>32.46</b>	<b>38.40</b>	<b>49.08</b>

$\sigma$	5	10	15	25
BPFA	40.37	<b>36.24</b>	<b>33.98</b>	<b>31.19</b>
CDF-KFA	39.94	35.63	33.52	30.60
KFA	<b>40.58</b>	36.20	33.73	30.54

**3-way data:** Here we show the superior inpainting performance of KFA on two 3-way data sets. The first is the amino acids fluorescence data ( $5 \times 201 \times 61$ ) (Bro, 1998), with  $5 \times 50 \times 4$  patches,  $\Delta = 1$ ,  $K = 256$ , and  $R = R_G = 20$ . We compare to BPFA, MGP-CP (no patches), and sBTD (Zhao et al., 2015). The second is the brainweb, a 3D MRI image (Collins et al., 1998), where we use  $8 \times 8 \times 8$  patches,  $\Delta = 4$ ,  $K = 512$ , and  $R = R_E = 24$ . For the brainweb data, we compare to BPFA, and sBTD. We also show comparable performance to BM4D for denoising (Maggioni et al., 2013). The results are shown in Tables 4–5. We report PSNR and relative residual square error (RRSE):  $\|\hat{\mathcal{X}} - \mathcal{X}\|_F / \|\mathcal{X}\|_F$ .

After reviewing the results for grayscale and color image inpainting, the substantial performance gain by KFA over the other tensor methods is not surprising and is due primarily to the fact that we are performing tensor dictionary learning, not tensor decomposition—

in fact, BPFA is comparable to tensor decomposition. For 20% pixels, KFA obtains a PSNR of 27.92, 29.58, and 30.46 dB, for gray, color, and MRI data. The reason inpainting performance increases as the dimension grows is that the number of sensed pixels in a patch increases with dimension (about 13 pixels for  $8 \times 8$ , and 102 for  $8 \times 8 \times 8$ ). We further note that for the amino experiment, KFA is rank-overcomplete, but not overcomplete ( $R_G < K < \prod m_t$ ).

Table 4: Reconstruction quality for amino 10% pixels.

	sBTD	MGP-CP	BPFA	KFA
RRSE	0.026	0.02542	0.02752	<b>0.01099</b>
PSNR	45.40	45.60	44.91	<b>52.88</b>

Table 5: Reconstruction PSNR (dB) for 3D MRI.

	sBTD	BM4D	BPFA	KFA
$\sigma = 19\%$	—	<b>29.70</b>	29.26	29.67
20% PIXELS	22.33	—	30.21	<b>30.46</b>

## 6.2 Deep learning

In the next two experiments we test deep CKFA and Kruskal DGDN. The rank for these experiments was set to  $R = 1$  to minimize training time. Even with such a strong restriction the proposed tensor models outperform the non-tensor models. The hyperparameters for our deep models are set as in (Pu et al., 2016a); no tuning or optimization was performed. All of our deep models are rank-overcomplete (each layer).

**Caltech 101:** We resize the images to  $128 \times 128$ , followed by local contrast normalization Jarrett et al. (2009). The network in this example has 3 layers. The dictionary sizes for each layer are set to  $K_1 = 64$ ,  $K_2 = 125$  and  $K_3 = 128$ , and the dictionary atom sizes are set to  $16 \times 16$ ,  $9 \times 9$  and  $5 \times 5$ . The size of the pooling regions are  $4 \times 4$  (layer 1 to layer 2) and  $2 \times 2$  (layer 2 to layer 3). For classification, we follow the setup in Yang et al. (2009), selecting 15 and 30 images per category for training, and up to 50 images per category for testing. The training for the CFA and CKFA models is unsupervised, and the top layer features are used to train an SVM after training the deep model.

Table 6: Caltech 101 classification accuracy (%).

METHOD	15 IMAGES	30 IMAGES
HBP-CFA LAYER-1	53.6	62.5
CKFA LAYER-1	55.7	63.2
DEEP CFA	43.24	53.57
HBP-CFA LAYER-1+2	58.0	65.7
CKFA LAYER-2	57.4	64.5
CKFA LAYER-3	61.2	69.9
DGDN	75.4	87.8
KRUSKAL DGDN	<b>76.1</b>	<b>88.3</b>

We compare to the hierarchical beta process CFA (HBP-CFA) (Chen et al., 2011), the pretrained DGDN (*i.e.*, deep CFA), and the DGDN (Pu et al., 2016a). The DGDN models have 3-layers. Other deep CNN models outperform our results in this experiment. However, we are simply showing that using a tensor structured dictionary can improve performance. The results for Caltech 101 are shown in Table 6. The one layer CKFA significantly outperforms Deep CFA and is also better than the single layer HBP-CFA. We also find that the 2-layer deep CKFA achieves comparable performance to the deep HBP-CFA using only the top-layer features. The 2-layer HBP-CFA uses both layers of features in the SVM. We also train a 3-layer CKFA and see that the deeper model continues to extract more discriminative features as layers are added. Kruskal DGDN provides state of the art accuracy among models *not* pretrained on ImageNet (Deng et al., 2009).

**MNIST:** A two layer model is used with dictionary size  $8 \times 8$  in the first layer and  $6 \times 6$  in the second layer; the pooling size is  $3 \times 3$  and the number of dictionary atoms in layers 1 and 2 are  $K_1 = 36$  and  $K_2 = 152$ . These numbers of dictionary atoms are obtained by setting the initial number of atoms to a large value ( $K_1 = 50$ ,  $K_2 = 200$ ), then removing low-use atoms.

The results for MNIST are shown in Table 7. The state of the art MCDNN is shown as a reference point, MCDNN uses a committee of 35 convnets, elastic distortions, and width normalization. In contrast, we train a single 2-layer model using the original training data.

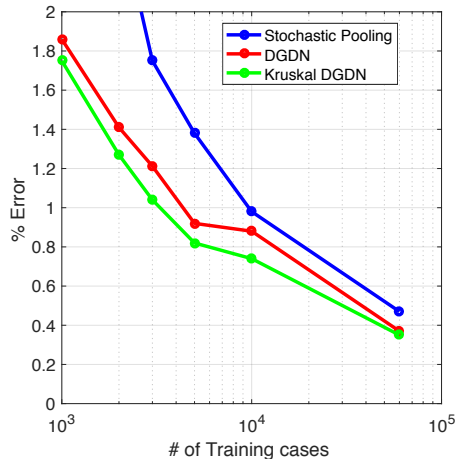


Figure 5: MNIST classification error varying training size.

The separable Kruskal DGDN outperforms DGDN. We also test the robustness of Kruskal DGDN to small data sets. The results are shown in Figure 5. We compare to the first stochastic pooling work (Zeiler and Fergus, 2013) and DGDN. Kruskal DGDN is consistently better than DGDN.

Table 7: MNIST classification error on test-set.

METHOD	ERROR %
35-CNN (CIRESAN ET AL., 2012)	0.23
6-LAYER CNN (CIRESAN ET AL., 2011)	0.35
4-LAYER CNN (LI ET AL.)	0.37
2-LAYER DGDN (PU ET AL., 2016A)	0.37
2-LAYER KRUSKAL DGDN	0.35

## 7 Conclusion

This paper presented a doubly nonparametric tensor-variate factor analysis model. To the best of our knowledge, this is the first factor analysis model capable of preserving the structure of arbitrary order tensor data. KFA is fully locally conjugate, by design, and can be implemented as a Gibbs sampler. The concept of rank-overcompleteness was also introduced. We showed that rank has a role in the interaction between sparsity and overcompleteness, but the interplay of these three properties requires further study. We also explored several extensions of KFA: online learning with BCDF, convolutional factor analysis, deep CFA, and supervised deep CFA using the Bayesian SVM. The performance of KFA and Kruskal DGDN is promising and has improved state of the art in inpainting and on the Caltech 101 image classification benchmark.

**Acknowledgments:** Supported by the Chemical Imaging, Signature Discovery, and Analytics in Motion Initiatives at PNNL and by ARO, DARPA, DOE, NGA, ONR and NSF. PNNL is operated by Battelle Memorial Inst. for the US DOE; contract DE-AC05-76RL01830.



## References

- H. Abo, G. Ottaviani, and C. Peterson. Induction for secant varieties of segre varieties. *Transactions of the American Mathematical Society*, 2009.
- M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE T Signal Process.*, 2006.
- A. Bhattacharya and D. Dunson. Sparse Bayesian infinite factor models. *Biometrika*, 2011.
- R. Bro. *Multi-way analysis in the food industry: models, algorithms, and applications*. PhD thesis, Københavns Universitet, 1998.
- B. Chen, G. Polatkan, G. Sapiro, D. Dunson, and L. Carin. The hierarchical beta process for convolutional factor analysis and deep learning. In *ICML*, 2011.
- O. Christensen. *Frames and bases: An introductory course*. Springer, 2008.
- A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Proc. Mag.*, 2015.
- D. Ciresan, J. Meier, U. and Masci, L. Maria Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI*, 2011.
- D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, 2012.
- D. Collins, A. Zijdenbos, V. Kollokian, J. Sled, N. Kabani, C. Holmes, and A. Evans. Design and construction of a realistic digital brain phantom. *IEEE T Med. Imaging*, 1998.
- P. Comon. Independent component analysis, a new concept? *Signal processing*, 36(3), 1994.
- P. Comon, G. Golub, L. Lim, and B. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM J Matrix Anal. A*, 30(3), 2008.
- J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- R. Gribonval, R. Jenatton, F. Bach, M. Kleinstueber, and M. Seibert. Sample complexity of dictionary learning and other matrix factorizations. *IEEE T Inform. Theory*, 2015.
- R. Guhaniyogi, S. Qamar, and D. B. Dunson. Bayesian conditional density filtering. *arXiv:1401.3632*, 2014.
- S. Hawe, M. Seibert, and M. Kleinstueber. Separable dictionary learning. In *CVPR*, 2013.
- K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.
- L. Kovarik, A. Stevens, A. Liyu, and N. Browning. Implementing an accurate and rapid sparse sampling approach for low-dose atomic resolution STEM imaging. *Appl. Phys. Lett.*, 109(16), 2016.
- M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural computation*, 2000.
- C. Li, A. Stevens, C. Chen, Y. Pu, Z. Gan, and L. Carin. Learning weight uncertainty with stochastic gradient MCMC for shape classification. In *CVPR*.
- M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE T Image Process.*, 2013.
- J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *arXiv:1411.3230*, 2014.
- J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *ICML*, 2009.
- N. Polson and S. Scott. Data augmentation for support vector machines. *Bayesian Analysis*, 2011.
- Y. Pu, X. Yuan, and L. Carin. Generative deep deconvolutional learning. *arXiv:1412.6039*, 2014.
- Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In *NIPS*, 2016a.
- Y. Pu, X. Yuan, A. Stevens, C. Li, and L. Carin. A deep generative deconvolutional image model. In *AISTATS*, 2016b.
- N. Qi, Y. Shi, X. Sun, and B. Yin. TenSR: Multi-dimensional tensor sparse representation. In *CVPR*, 2016.
- P. Rai, Y. Wang, S. Guo, G. Chen, D. Dunson, and L. Carin. Scalable bayesian low-rank decomposition of incomplete multiway tensors. In *ICML*, 2014.
- A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit, and P. Fua. Learning separable filters. *IEEE T Pattern Anal.*, 2015.
- A. Stevens, H. Yang, L. Carin, I. Arslan, and N. Browning. The potential for Bayesian compressive sensing to significantly reduce electron dose in high-resolution STEM images. *Microscopy*, 63(1), 2014.
- Y. Tang, R. Salakhutdinov, and G. Hinton. Tensor analyzers. In *ICML*, 2013.
- Y. Wang and L. Carin. Levy measure decompositions for the beta and gamma processes. In *ICML*, 2012.
- Z. Xing, M. Zhou, A. Castrodad, G. Sapiro, and L. Carin. Dictionary learning for noisy and incomplete hyperspectral images. *SIAM J. Imag. Sci.*, 2012.
- J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.
- X. Yuan, V. Rao, S. Han, and L. Carin. Hierarchical infinite divisibility for multiscale shrinkage. *IEEE T Signal Process.*, 2014.
- M. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.
- X. Zhang, D. Dunson, and L. Carin. Tree-structured infinite sparse factor model. In *ICML*, 2011.
- Y. Zhang, R. Henao, C. Li, and L. Carin. Bayesian dictionary learning with gaussian processes and sigmoid belief networks. In *IJCAI*, 2016.
- Q. Zhao, L. Zhang, and A. Cichocki. Bayesian sparse Tucker models for dimension reduction and tensor completion. *arXiv:1505.02343*, 2015.
- M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric Bayesian dictionary learning for analysis of noisy and incomplete images. *IEEE T Image Process.*, 2012.