

Tensor Methods for Generating Compact Uncertainty Quantification and Deep Learning Models

(Invited Special Session Paper)

Chunfeng Cui^{*†}, Cole Hawkins^{*‡}, and Zheng Zhang[†]

[†]Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

[‡]Department of Mathematics, University of California, Santa Barbara, CA 93106

Abstract—Tensor methods have become a promising tool to solve high-dimensional problems in the big data era. By exploiting possible low-rank tensor factorization, many high-dimensional model-based or data-driven problems can be solved to facilitate decision making or machine learning. In this paper, we summarize the recent applications of tensor computation in obtaining compact models for uncertainty quantification and deep learning. In uncertainty analysis where obtaining data samples is expensive, we show how tensor methods can significantly reduce the simulation or measurement cost. To enable the deployment of deep learning on resource-constrained hardware platforms, tensor methods can be used to significantly compress an over-parameterized neural network model or directly train a small-size model from scratch via optimization or statistical techniques. Recent Bayesian tensorized neural networks can automatically determine their tensor ranks in the training process.

I. INTRODUCTION

As an efficient tool to overcome the curse of dimensionality, tensor decomposition methods date back to 1927 [1] and have been employed in many application fields such as computer vision [2], signal processing [3], [4], graph matching [5], bio-informatics [6], etc. Different from its matrix counterpart (i.e., singular value decomposition), tensor decompositions have different formats, such as the CP decomposition [1], Tucker decomposition [7], tensor-train decomposition [8], tensor network factorization [9], [10], t-SVD decomposition [11], and so forth. Some papers have provided excellent surveys of tensor computation and its applications [4], [12].

This paper will provide a high-level survey of tensor computation in the following two application fields: uncertainty-aware design automation and deep learning. These two seemingly irrelevant topics both require compact computational models to facilitate their subsequent statistical estimation, performance prediction and hardware implementation, despite their fundamentally different challenges:

- An EDA framework involves many modeling, simulation, and optimization modules. These modules often require some model-based simulation or hardware measurement data to decide the next step, but obtaining each piece of data is expensive. This challenge becomes more significant as

process variations increase: one needs more data to capture an uncertain performance space. Therefore, it is desirable to extract high-quality “compact” models to facilitate a decision making process with a “small” available data set.

- In deep learning, “big” training data sets are often easy to obtain, and large-size neural networks can be trained on powerful platforms (e.g., in the cloud or on local high-performance servers). However, deploying them on resource-constrained hardware platforms (e.g., embedded systems and IoT devices) becomes a big challenge. As a result, there is a strong motivation to develop compact neural network models that can be deployed with low memory and computational cost.

For both applications, tensor methods can be used to develop compact models with low computational and memory cost.

II. TENSOR DECOMPOSITION AND COMPLETION

We first give a high-level tutorial about two important tensor problems: tensor decomposition and tensor completion. The first is often used to generate a compact low-rank representation when a (big) complete data set is given. The second is often employed when a (small) portion of the data is available.

Fig. 1 shows a tensor and several popular tensor decomposition techniques. A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_d}$ is a d -dimensional data array with $d \geq 3$. It reduces to a matrix when $d = 2$ and a vector when $d = 1$. The high dimensionality of a tensor often brings in higher expressive power and higher compression capability. Three mainstream tensor decomposition methods are widely used for data analysis, scientific computing, and machine learning:

- The **CP decomposition** [1] method decomposes a tensor into the summation of R rank-one terms:

$$\mathcal{A} \approx \sum_{r=1}^R \mathbf{x}_r^{(1)} \circ \mathbf{x}_r^{(2)} \circ \dots \circ \mathbf{x}_r^{(d)} \Leftrightarrow a_{i_1 i_2 \dots i_d} \approx \sum_{r=1}^R \prod_{k=1}^d \mathbf{x}_r^{(k)}(i_k).$$

Here $\circ$ denotes the outer product and $\mathbf{x}_r^{(k)} \in \mathbb{R}^{I_k}$, and $a_{i_1 i_2 \dots i_d}$ denotes the scalar element in $\mathcal{A}$ indexed by $(i_1, i_2, \dots, i_d)$. The total storage complexity is reduced to $\sum_{i=1}^d I_i R$. When the approximation is replaced with equality, the minimal integer of R is called the *tensor rank*.

^{*}Equally contributed authors. This work was partly supported by NSF CAREER Award No. 1846476, NSF-CCF Awards No. 1763699 and No. 1817037, and an UCSB start-up grant.

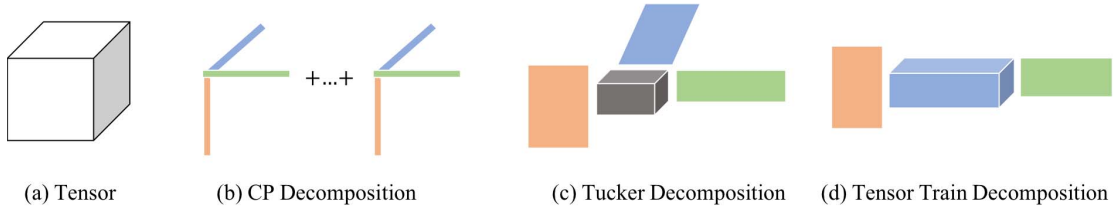


Fig. 1. Several popular tensor compositions.

- The **Tucker decomposition** [7] compresses a tensor into a smaller core tensor $\mathcal{G}$ and d orthogonal factor matrices $\{\mathbf{U}_k \in \mathbb{R}^{I_k \times R_k}\}_{k=1}^d$:

$$a_{i_1 \dots i_d} = \sum_{r_1, \dots, r_d=1}^{R_1, \dots, R_d} g_{r_1 \dots r_d} \mathbf{U}_1(i_1, r_1) \dots \mathbf{U}_d(i_d, r_d). \quad (1)$$

The Tucker rank is bounded by $R_i \leq I_i$ for all $i = 1, \dots, d$. The storage complexity is reduced to $\sum_{i=1}^d I_i R_i + \prod_{i=1}^d R_i$.

- The **tensor-train decomposition** [8] writes a tensor as a series of three-dimensional factor tensors, i.e.,

$$a_{i_1 \dots i_d} = \mathcal{G}_1(:, i_1, :) \mathcal{G}_2(:, i_2, :) \dots \mathcal{G}_d(:, i_d, :). \quad (2)$$

Here $\mathcal{G}_k \in \mathbb{R}^{R_{k-1} \times I_k \times R_k}$, $R_0 = R_d = 1$, and $\mathcal{G}_k(:, i_k, :)$ is a MATLAB-like expression for the i_k -th lateral slice of $\mathcal{G}_k$. For a given tensor-train rank $\mathbf{R} = (R_0, R_1, \dots, R_d)$, the storage complexity is reduced to $\sum_{k=1}^d I_k R_{k-1} R_k$.

Tensor Completion. Given only partial elements of a tensor, the tensor completion or tensor recovery problem solves

$$\min_{\mathcal{A} \in \mathcal{M}} \|P_\Omega(\mathcal{A} - \mathcal{X})\|_F^2, \quad (3)$$

where $\mathcal{M}$ denotes the set of low rank tensors in a proper format (e.g., CP, Tucker or tensor-train format), the projection $P_\Omega(\mathcal{A})$ keeps the element $a_{i_1 i_2 \dots i_d}$ for all $(i_1, \dots, i_d) \in \Omega$ and sets other elements to zero. The Frobenius norm is defined as $\|\mathcal{A}\|_F^2 = \sum_{i_1, \dots, i_d} a_{i_1 \dots i_d}^2$. The cost function and regularization may be modified dependent on practical applications.

III. TENSORS FOR UNCERTAINTY/VARIABILITY ANALYSIS

A. Data-Expensive EDA Problems

EDA problems are often model-driven and data-expensive: the design problems are well described by a detailed mathematical model (e.g., Maxwell equation for interconnect or RF device modeling, modified nodal analysis for circuit simulation), and one often needs to solve such an expensive mathematical model repeatedly or iteratively to get enough data (e.g., gradient information) to decide the next step (e.g., to optimize a circuit design parameter). The involved numerical computation makes the data acquisition expensive. To accelerate the whole data-expensive EDA flow, one can choose to:

- Speed up the acquisition of each data sample. Representative matrix/vector-based techniques include fast PDE solvers [13]–[15], fast circuit simulators [16], [17] and

model order reduction techniques [18]–[20]. These solvers are often deterministic.

- Reduce the number of data acquisitions. This is especially important for nano-scale design that is highly influenced by process variations. In this case, one needs a huge amount of data samples to characterize the uncertain circuit performance. Representative techniques include fast Monte Carlo [21] and recent stochastic spectral methods [22]–[24].

Tensor techniques can be employed to achieve both goals. Representative tensor techniques for the first goal include [25]–[27]. In this paper, we focus on tensor techniques for uncertainty and variability analysis as summarized in Table I, and we elaborate their key ideas below.

B. Tensor Methods for Uncertainty Propagation

Uncertainty quantification techniques aim to predict and control the probability density function (PDF) of the system output $y(\xi)$ under some random parameters $\xi \in \mathbb{R}^d$ describing process variations. The stochastic spectral methods based on generalized polynomial chaos method [28] have significantly outperformed Monte Carlo in many application domains. The key idea is to approximate $y(\xi)$ as a truncated linear combination of some specialized orthogonal basis functions $\{\Psi_\alpha(\xi)\}$ of ξ . The weight of each basis can be computed by various numerical techniques such as stochastic collocation [29], stochastic Galerkin [30] and stochastic testing [22]. When the parameters are non-Gaussian correlated, one can also employ the modified basis functions and stochastic collocation methods proposed in [31]–[34].

Curse of Dimensionality. Stochastic spectral methods suffer from an extremely high computational cost as the number of random parameters increase. For instance, in sampling-based techniques, the number of simulation samples may increase exponentially as d increases. Some tensor solvers have been developed to address this fundamental challenge:

- **Tensor-Based Stochastic Collocation** [35], [36]. The stochastic collocation method uses a projection method to compute the weight of each basis. Standard techniques discretize each random parameter into m points, leading to m^d simulation samples in total. Instead of simulating all samples, the technique in [36] only simulates a small number of random samples and estimate the big unknown simulation data set by a tensor completion subject to two constraints: (1) the recovered tensor is low-rank; (2) the

TABLE I
APPLICATION OF TENSORS IN UNCERTAINTY PROPAGATION AND VARIABILITY MODELING

Reference	Problem	Key Idea
[35], [36], [38], [39]	high-dim stochastic collocation	tensor completion to estimate unknown simulation data
[37]	hierarchical uncertainty quantification	tensor-train decomposition for high-dim integration
[33]	uncertainty analysis with non-Gaussian correlated uncertainty	functional tensor train to compute basis functions
[40]	spatial variation pattern prediction	statistical tensor completion to predict variation pattern

resulting generalized polynomial chaos expansion is sparse. This technique has been successfully applied to electronic IC, photonics and MEMS with up to 57 random parameters.

- **Tensor-Based Hierarchical Uncertainty Analysis [37].** Hierarchical techniques can be used to analyze the uncertainty of a complex system consisting of multiple interconnected components or subsystems. The key idea is to simulate each subsystem by a fast stochastic spectral method, then use their outputs as new random inputs for the system-level configuration. A major computational bottleneck is the high-dimensional integration required to recompute the basis function at the system level. In [37] a tensor train decomposition is used to reduce the repetitive functional evaluation cost from an exponential cost to a linear one. This technique has enabled efficient uncertainty analysis of a MEMS/IC co-design with 184 process variation parameters.
- **Tensor Method to Handle Non-Gaussian Correlated Uncertainties [33].** A fundamental challenge of uncertainty propagation is how to handle non-Gaussian correlated process variations. Recently a set of basis functions and stochastic collocation methods were developed by [31], [32] to achieve high accuracy and efficiency. It is expensive to compute the basis functions in a high-dimensional case. When the non-Gaussian correlated random parameters are described by a Gaussian mixture density function, the basis functions were efficiently calculated by a functional tensor train method [33], [34]. The key idea is as follows: the integration of a d -variable polynomial over each correlated Gaussian density can be written as the product of moments for each random variable.

C. Tensor Methods in Variability Prediction

Statistical simulation of a circuit or device requires a given detailed statistical description (e.g., a probability density function) of the process variations. These statistical models are normally obtained by measuring the performance data of a huge number of testing chips. However, measuring the testing chips costs time and money, and may also cause mechanical damage. Existing techniques such as the virtual probe [41] use the compressed sensing technique to predict the 2-D spatial variation pattern based on limited data.

In our recent paper [40], we proposed to simultaneously predict the variation patterns of multiple dies. If each die has $N_1 \times N_2$ devices to test, we can stack N_3 dies together to form a tensor. Then, we employed the Bayesian tensor completion technique in [42] to predict the spatial variation pattern with only a small number of testing samples. This technique can automatically determine the tensor rank, achieving around 0.5%

relative errors with only 10% testing samples with significant memory and computational cost reduction compared with the virtual probe technique [41].

IV. COMPACT DEEP LEARNING MODELS

Different from model-driven and data-expensive EDA problems, deep learning is suitable for data-driven and data-cheap applications such as computer vision and speech recognition. With the huge amount of available data (e.g., obtained from the social networks and many edge devices) and today's powerful computing platforms, deep learning has achieved success in a wide range of practical applications. However, deploying large neural networks requires huge computational and memory resource, limiting their applications on resource-constrained devices (i.e. smartphones, mobile robotics). Therefore, it is highly desirable to build compact neural network models that can be deployed with low hardware cost.

Many techniques can help generating more compact deep learning models. Most existing techniques are applied to individual weights, convolution filters or neurons, for instance:

- **Pruning [43]:** the key idea is to generate a sparse deep neural network by removing some redundant neurons that are not sensitive to the prediction performance.
- **Quantization [44]:** considering that model parameters are actually represented with binary bits in hardware, one may reduce the number of bits with little loss of accuracy.
- **Knowledge Distillation [45]:** the key idea is to shift the information from a deep and wide (teacher) neural network to a shallow one (i.e., a student network).
- **Low-rank Compression [46]:** one can also compress the weight matrix or convolution filters by low-rank matrix or tensor decomposition.

This section will survey the recent low-rank tensor techniques for generating compact deep neural network models. These techniques can be classified into two broad families:

- **Tensorized Inference:** these techniques employ a “train-then-compress” flow. Firstly a large deep neural network is trained (possibly with a GPU cluster), then tensor decomposition is applied to compress this pre-trained model to enable its deployment on a hardware platform with limited resources (e.g., on a smart phone).
- **Tensorized Training:** these techniques skip the expensive training on a high-performance platform, and they aim to directly train a compact tensorized neural network from scratch and in an end-to-end manner.

A common challenge of the above technique is to determine the tensor rank. Exactly determining a tensor rank in general

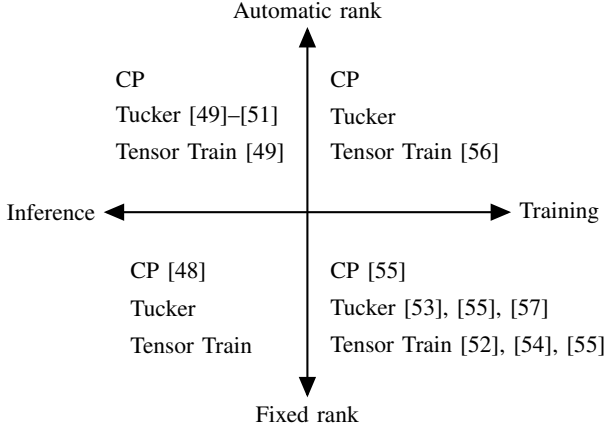


Fig. 2. Summarize existing tensorized deep neural networks.

is NP-hard [47]. Therefore, in practice one often leverages numerical optimization or statistical techniques to obtain a reasonable rank estimation. Dependent on the capability of automatic rank determination, existing tensorized neural network methods can be further classified into four groups as shown in Fig. 2. We will elaborate their key ideas below.

A. Tensorized Inference with a Fixed Rank

Lebedev et al. [48] firstly applied CP tensor factorization to compress large-scale neural networks with fully connected layers. Because it is hard to automatically determine the exact CP tensor rank, this method keeps the tensor rank fixed in advance and employs an alternating least square method to compress folded weight matrices.

B. Tensorized Inference with Automatic Rank Determination

Compared with CP factorization, Tucker and tensor-train decompositions allow one to adjust the ranks based on accuracy requirement, and they have been employed in [49], [50] to compress both fully connected layers and convolution layers. A hardware prototype was even demonstrated for mobile applications in [50]. Recently, an iterative compression technique was further developed to improve the compression ratio and model accuracy [51].

C. Tensorized Training with a Fixed Rank

In order to avoid the expensive pre-training in the uncompressed format, the work in [52] and [53] directly trained fully connected and convolution layers in low-rank tensor-train and Tucker format with the tensor ranks fixed in advance. This idea has also been applied to recurrent neural networks [54], [55].

In the following, we would like to summarize the storage and computational complexity (e.g. flops) by applying CP decomposition, Tucker decomposition, and tensor-train decomposition for both the convolution layers and the fully connected layers:

- **Tensorized Convolution Layers.** Consider a convolutional weight $\mathcal{K} \in \mathbb{R}^{l \times l \times C \times S}$, where l is the filter size, and C, S

are the number of input and output channels respectively. This kernel requires l^2CS parameters in total. The tensor-train format reduces the number of parameters from l^2CS to $O(R^2(2l + C + S))$. The rank- R CP-decomposition can represent the weight with $O(R(2l + C + S))$ storage. The Tucker-2 decomposition [57] only compresses the input channel and output channel dimension, and requires $O(R(Rl^2 + C + S))$ storage. We also note that in practice it is common to reshape the tensor $\mathcal{K}$ into a higher-order tensor [as shown in Fig. 3 (a)]. Specifically, one can factorize

$$C = \prod_{k=1}^d c_k \text{ and } S = \prod_{k=1}^d s_k$$

and reformulate $\mathcal{K}$ as a $(2d + 2)$ -dimensional tensor $\mathcal{K} \in \mathbb{R}^{l \times l \times c_1 \times \dots \times c_d \times s_1 \times \dots \times s_d}$. Such folding can often result in a higher compression ratio.

- **Tensorized Fully Connected Layers.** The weight in the fully connected layer is a 2D-matrix, and direct application of the tensor method reduces to a matrix singular value decomposition. Given a weight matrix $\mathbf{W} \in \mathbb{R}^{M \times N}$ one can factorize $M = \prod_{k=1}^d M_k$, $N = \prod_{k=1}^d N_k$ and represent $\mathbf{W}$ with a high-dimensional tensor $\mathcal{W} \in \mathbb{R}^{m_1 \times n_1 \times \dots \times m_d \times n_d}$ [as shown in Fig. 3 (b)]. Then by the tensor factorization, the number of parameters can reduce from $M \times N$ to $O(r(md + nd))$, and $O(dr^2mn)$ for a CP format and tensor-train format, respectively.

We summarize both the storage and computational complexity of different tensor compression methods in Table II. For the convolutional layer, we only counts the computational costs of a $l \times l \times C$ block.

D. Tensorized Training with Automatic Rank Determination

It is a challenging task to automatically determine the tensor rank in a training process due to the following reasons:

- Different from the matrix case, there is not a proper surrogate model for the rank of a high-order tensor. As a result, it is non-trivial to regularize the loss function of a neural network with a low-rank penalty term.
- Existing tensor decomposition methods work on a given full tensor. However, the tensors in end-to-end training are embedded within a deep neural network in a highly nonlinear manner. This also makes existing tensor completion or recovery frameworks fail to work.

In order to address this fundamental challenge and to enable efficient end-to-end training, we leveraged the variational inference and proposed the first Bayesian tensorized neural network [56] to automatically determine the tensor rank as part of the training process. We use the notation $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ to represent the training data. Our goal is to learn the low-rank tensor parameters by estimating the following posterior density:

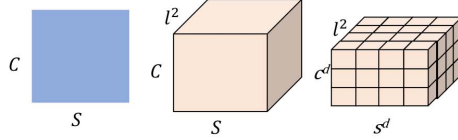
$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \quad (4)$$

where $\boldsymbol{\theta}$ include the unknown tensor-train factors and some hyper-parameters, and $p(\boldsymbol{\theta})$ is a prior density to enforce

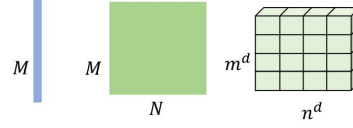
TABLE II

SPEEDUP OF STORAGE AND COMPUTATIONAL COMPLEXITY OF TENSOR METHODS FOR THE CONVOLUTIONAL (CONV) LAYER AND THE FULLY CONNECTED (FC) LAYER. ‘-’ DENOTES INAPPLICABLE: TUCKER DECOMPOSITION IS NOT APPLICABLE TO HIGH-ORDER TENSORS DUE TO THE CURSE OF DIMENSIONALITY; FC WEIGHT IS A 2D MATRIX.

	Original			Tensor decomposition		High-order tensor decomposition	
	Storage	FLOPS		Storage	FLOPS	Storage	FLOPS
Conv	$O(CSl^2)$	$O(CSl^2)$	CP	$O(R(2l + C + S))$	$O(R(Cl^2 + S))$	$O(r(2l + cd + sd))$	$O(r(Cl^2 + S))$
			Tucker	$O(R(Rl^2 + C + S))$	$O(R(Cl^2 + Rl^2 + S))$	-	-
			TT	$O(R^2(2l + C + S))$	$O(R^2(2Cl^2 + C + S))$	$O(r^2(2l + dc + ds))$	$O(r^2(2Cl^2 + dC + dS))$
FC	$O(MN)$	$O(MN)$	CP	$O(R(C + S))$	$O(R(C + S))$	$O(dmr + dnr)$	$O(Mr + Nr)$
			Tucker	-	-	-	-
			TT	-	-	$O(r^2dmn)$	$O(r^2md \max(M, N))$



(a) Convolutional layer



(b) Fully connected layer

Fig. 3. (a) The convolutional kernel can be regarded as a 3D tensor in $C \times S \times l^2$ or a 4D tensor in $C \times S \times l \times l$. We can also reformulate it into a $(2d + 2)$ -dimensional tensor. (b) The weight matrix in the fully connected layer is a 2D matrix, which can be reformulated into a $2d$ -dimensional tensor.

low-rank property. The following key techniques enabled the efficient training and automatic rank determination:

- The prior density $p(\theta)$ is designed by considering the coupling of adjacent tensor-train cores, such that their ranks can be controlled simultaneously.
- We employed a Stein variational gradient descent [58] method to approximate the posterior density $p(\theta|\mathcal{D})$. This method combines the flexibility of Markov-Chain Monte Carlo and the efficiency of optimization techniques, which is beyond the capability of the mean-field inference framework in the Bayesian tensor completion framework [42].

This method has trained a two-layer fully connected neural network, a 6-layer CNN and a 110-layer residual neural network, leading to $7.4\times$ to $137\times$ compression ratios.

V. CONCLUSIONS

In this paper, we have revisited several compact models generated by the tensor decomposition/completion approach. For the data-expansive problems arising from EDA, we have summarized several tensor methods in uncertainty quantification and spatial prediction. Tensor techniques have successfully solved many high-dimensional uncertainty quantification problems with both independent and non-Gaussian random parameters. They have also significantly reduced the chip testing cost in spatial variation pattern prediction.

In the context of deep learning, tensor decomposition proves to be an efficient technique to obtain compact learning models. They have achieved significant compression in both inference and training. Our recent Bayesian tensorized neural network allows automatic tensor rank determination in the end-to-end training process.

REFERENCES

- [1] F. L. Hitchcock, “The expression of a tensor or a polyadic as a sum of products,” *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.
- [2] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 208–220, 2012.
- [3] 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 Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.
- [4] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, “Tensor decomposition for signal processing and machine learning,” *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.
- [5] C. Cui, Q. Li, L. Qi, and H. Yan, “A quadratic penalty method for hypergraph matching,” *Journal of Global Optimization*, vol. 70, no. 1, pp. 237–259, 2018.
- [6] T. J. Durham, M. W. Libbrecht, J. J. Howbert, J. Bilmes, and W. S. Noble, “Predictd parallel epigenomics data imputation with cloud-based tensor decomposition,” *Nature communications*, vol. 9, no. 1, p. 1402, 2018.
- [7] L. R. Tucker, “Some mathematical notes on three-mode factor analysis,” *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.
- [8] I. V. Oseledets, “Tensor-train decomposition,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.
- [9] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic *et al.*, “Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions,” *Foundations and Trends® in Machine Learning*, vol. 9, no. 4-5, pp. 249–429, 2016.
- [10] A. Cichocki, A.-H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D. P. Mandic *et al.*, “Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives,” *Foundations and Trends® in Machine Learning*, vol. 9, no. 6, pp. 431–673, 2017.
- [11] Z. Zhang and S. Aeron, “Exact tensor completion using t-svd,” *IEEE Transactions on Signal Processing*, vol. 65, no. 6, pp. 1511–1526, 2016.
- [12] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.
- [13] M. Kamon, M. J. Tsuk, and J. K. White, “Fasthenry: A multipole-accelerated 3-d inductance extraction program,” *IEEE Transactions on Microwave theory and techniques*, vol. 42, no. 9, pp. 1750–1758, 1994.

- [14] K. Nabors and J. White, "Fastcap: A multipole accelerated 3-d capacitance extraction program," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1447–1459, 1991.
- [15] J. R. Phillips and J. K. White, "A precorrected-fft method for electrostatic analysis of complicated 3-d structures," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059–1072, 1997.
- [16] K. S. Kundert, J. K. White, and A. L. Sangiovanni-Vincentelli, *Steady-state methods for simulating analog and microwave circuits*, 2013, vol. 94.
- [17] P. Li *et al.*, "Parallel circuit simulation: A historical perspective and recent developments," *Foundations and Trends® in Electronic Design Automation*, vol. 5, no. 4, pp. 211–318, 2012.
- [18] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 22, no. 2, pp. 155–170, 2003.
- [19] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White, "A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 678–693, 2004.
- [20] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: passive reduced-order interconnect macromodeling algorithm," in *Proc. Intl. Conf. Computer-aided design*, 1997, pp. 58–65.
- [21] A. Singhee, S. Singhal, and R. A. Rutenbar, "Practical, fast monte carlo statistical static timing analysis: why and how," in *Proc. Int. Conf. Computer-Aided Design*, 2008, pp. 190–195.
- [22] Z. Zhang, T. A. El-Moselhy, I. A. M. Elfadel, and L. Daniel, "Stochastic testing method for transistor-level uncertainty quantification based on generalized polynomial chaos," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1533–1545, Oct. 2013.
- [23] Z. Zhang, X. Yang, G. Marucci, P. Maffezzoni, I. M. Elfadel, G. Karniadakis, and L. Daniel, "Stochastic testing simulator for integrated circuits and MEMS: Hierarchical and sparse techniques," in *Proc. IEEE Custom Integrated Circuits Conf.* San Jose, CA, Sept. 2014, pp. 1–8.
- [24] P. Manfredi, D. V. Ginste, D. D. Zutter, and F. Canavero, "Stochastic modeling of nonlinear circuits via SPICE-compatible spectral equivalents," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 61, no. 7, pp. 2057–2065, July 2014.
- [25] H. Liu, L. Daniel, and N. Wong, "Model reduction and simulation of nonlinear circuits via tensor decomposition," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1059–1069, 2015.
- [26] H. Liu, X. Y. Xiong, K. Batselier, L. Jiang, L. Daniel, and N. Wong, "STAVES: Speedy tensor-aided volterra-based electronic simulator," in *Proc. ICCAD*, 2015, pp. 583–588.
- [27] Z. Chen, S. Zheng, and V. I. Okhmatovski, "Tensor train accelerated solution of volume integral equation for 2-d scattering problems and magneto-quasi-static characterization of multiconductor transmission lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 6, pp. 2181–2196, 2019.
- [28] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM J. Sci. Comp.*, vol. 24, no. 2, pp. 619–644, Feb 2002.
- [29] D. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *SIAM Journal on Scientific Computing*, vol. 27, no. 3, pp. 1118–1139, 2005.
- [30] R. Ghanem and P. Spanos, *Stochastic finite elements: a spectral approach*. Springer-Verlag, 1991.
- [31] C. Cui, M. Gershman, and Z. Zhang, "Stochastic collocation with non-Gaussian correlated parameters via a new quadrature rule," in *Proc. IEEE Conf. EPEPS*. San Jose, CA, Oct. 2018, pp. 57–59.
- [32] C. Cui and Z. Zhang, "Stochastic collocation with non-Gaussian correlated process variations: Theory, algorithms and applications," *IEEE Trans. Components, Packag. and Manufacturing Tech.*, vol. 9, no. 7, pp. 1362 – 1375, July 2019.
- [33] —, "Uncertainty quantification of electronic and photonic ICs with non-Gaussian correlated process variations," in *Proc. Intl. Conf. Computer-Aided Design*. San Diego, CA, Nov. 2018, pp. 1–8.
- [34] —, "High-dimensional uncertainty quantification of electronic and photonic IC with non-Gaussian correlated process variations," *IEEE Trans. CAD of Integrated Circuits and Systems*, 2019.
- [35] Z. Zhang, T.-W. Weng, and L. Daniel, "A big-data approach to handle process variations: Uncertainty quantification by tensor recovery," in *IEEE Workshop on Signal and Power Integrity*, 2016, pp. 1–4.
- [36] —, "Big-data tensor recovery for high-dimensional uncertainty quantification of process variations," *IEEE Trans. Components, Packaging and Manufacturing Technology*, vol. 7, no. 5, pp. 687–697, 2017.
- [37] Z. Zhang, I. Oseledets, X. Yang, G. E. Karniadakis, and L. Daniel, "Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 34, no. 1, pp. 63 – 76, Jan 2015.
- [38] K. Konakli and B. Sudret, "Global sensitivity analysis using low-rank tensor approximations," *Reliability Engineering & System Safety*, vol. 156, pp. 64–83, 2016.
- [39] —, "Reliability analysis of high-dimensional models using low-rank tensor approximations," *Probabilistic Engineering Mechanics*, vol. 46, pp. 18–36, 2016.
- [40] J. Luan and Z. Zhang, "Prediction of multi-dimensional spatial variation data via bayesian tensor completion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.
- [41] W. Zhang, X. Li, F. Liu, E. Acar, R. A. Rutenbar, and R. D. Blanton, "Virtual probe: a statistical framework for low-cost silicon characterization of nanoscale integrated circuits," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 30, no. 12, pp. 1814–1827, 2011.
- [42] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian cp factorization of incomplete tensors with automatic rank determination," *IEEE trans. Pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1751–1763, 2015.
- [43] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Advances in neural information processing systems*, 1989, pp. 177–185.
- [44] G. Hinton and D. Van Camp, "Keeping neural networks simple by minimizing the description length of the weights," in *Proc. ACM Conf. on Computational Learning Theory*. Citeseer, 1993.
- [45] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [46] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *NIPS*, 2014, pp. 1269–1277.
- [47] J. Håstad, "Tensor rank is np-complete," *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.
- [48] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *Int. Conf. Learning Representations*, 2015.
- [49] T. Garipov, D. Podoprikin, A. Novikov, and D. Vetrov, "Ultimate tensorization: compressing convolutional and fc layers alike," *arXiv preprint arXiv:1611.03214*, 2016.
- [50] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.
- [51] J. Gusak, M. Kholavchenko, E. Ponomarev, L. Markeeva, I. Oseledets, and A. Cichocki, "One time is not enough: iterative tensor decomposition for neural network compression," *arXiv preprint arXiv:1903.09973*, 2019.
- [52] A. Novikov, D. Podoprikin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in neural information processing systems*, 2015, pp. 442–450.
- [53] G. G. Calvi, A. Moniri, M. Mahfouz, Z. Yu, Q. Zhao, and D. P. Mandic, "Tucker tensor layer in fully connected neural networks," *arXiv preprint arXiv:1903.06133*, 2019.
- [54] A. Tjandra, S. Sakti, and S. Nakamura, "Compressing recurrent neural network with tensor train," in *Int. Joint Conf. Neural Networks*, 2017, pp. 4451–4458.
- [55] —, "Tensor decomposition for compressing recurrent neural network," in *Int. Joint Conf. Neural Networks*, 2018, pp. 1–8.
- [56] C. Hawkins and Z. Zhang, "Bayesian tensorized neural networks with automatic rank selection," *arXiv preprint arXiv:1905.10478*, 2019.
- [57] J. Kossaiji, A. Khanna, Z. Lipton, T. Furlanello, and A. Anandkumar, "Tensor contraction layers for parsimonious deep nets," in *Proc. Computer Vision and Pattern Recognition Workshops*, 2017, pp. 26–32.
- [58] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," in *Advances in neural information processing systems*, 2016, pp. 2378–2386.