Bayesian Tensor Networks with Structured Posteriors

Kriton Konstantinidis, Yao Lei Xu, Danilo P. Mandic
Department of Electrical and Electronic Engineering
Imperial College London
{k.konstantinidis19,yao.xu15,d.mandic}@imperial.ac.uk

Qibin Zhao
Tensor Learning Team
RIKEN Center for Advanced Intelligence Project
qibin.zhao@riken.jp

Abstract

Tensor network (TN) methods have proven their considerable potential in deterministic regression and classification related paradigms, but remain underexplored in probabilistic settings. To this end, we introduce a variational inference TN framework for supervised learning, referred to as the Bayesian Tensor Network (BTN). This is achieved by making use of the multi-linear nature of tensor networks to construct a structured variational model which scales linearly with data dimensionality. The so imposed low rank structure on the tensor mean and Kronecker separability on the local covariances, make it possible to efficiently induce weight dependencies in the posterior distribution, thus enhancing model expressiveness at a drastically lower parameter complexity compared to the standard mean-field approach. A comprehensive validation of the proposed approach indicates the competitiveness of BTNs against modern structured Bayesian neural network approaches, while exhibiting enhanced interpretability and efficiency.

1 Introduction

Tensor networks (TNs) have been recently employed in a variety of Machine Learning (ML) paradigms. The increasing interest of the ML community in TNs stems from their suitability to operate with both dense and sparse data, their Big Data compatible properties, such as linear scaling with data dimensionality and constant scaling with the training size (assuming mini-batch gradient descent) [1], as well as their enhanced interpretability, owing to their multi-linear nature. The applicability and potential of TNs have been demonstrated through state-of-the-art performance in tasks including multi-modal learning [2], compression of large-dimensional data [3], sequence to sequence learning [4], anomaly detection [5] and theoretical analysis of neural networks [6], to name but a few. However, a vast majority of those achievements have been in deterministic settings, yet modern ML can greatly benefit from probabilistic tools. To fill this void, motivated by the appealing properties of TNs, we introduce the Bayesian Tensor Network (BTN) model, a variational inference TN framework for probabilistic supervised learning.

Contributions: The contributions of this work are threefold. First, we present a basic Mean-Field Variational Inference approach for TN based supervised learning. Second, by taking advantage of the multi-linear properties of TNs, we introduce a structured approach, where the learnable parameters are not treated independently, but as a realization of a tensor variate Gaussian random variable [7]. We demonstrate that this interpretation allows for an efficient information sharing in the posterior distribution. This is imposed by a Kronecker structure on the tensor mean and covariances, making

Second Workshop on Quantum Tensor Networks in Machine Learning, 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

their forms more flexible while simultaneously allowing for more accurate predictions, all at a significantly reduced parameter complexity compared to the mean-field case. The advantages of this novel framework are demonstrated through an example of a credibility interval construction based on the multi-linear properties of TNs. We believe that this approach, coupled with domain-dependent and physically relevant feature maps (basis functions), promises to provide meaningful model interpretations without sacrificing model expressiveness.

Related Work: An exhaustive review of structured approaches for variational Bayesian neural networks (BNNs) is out of the scope of this paper. However, we find it important to highlight the following works which exhibit the closest relationships to our approach. The idea of treating model parameters as a realization from a higher-order distribution in the context of variational Bayesian deep learning was first introduced in [8], where the authors considered a structured matrix-variate Gaussian posterior to model the posterior distribution of a BNN. Whereas their focus was on neural networks, we consider models which are fully described in a TN form, which allows for a physically meaningful treatment of the model parameters as a realization of a tensor-variate Gaussian posterior. Another related work [9] develops a tensor-variate Gaussian Process Prior Variational Autoencoder, in the context of learning latent representations, different from our focus on regression tasks. Furthermore, the work in [9] considers neural networks and imposes a tensor structure on the latent space, while our approach is concerned with a TN model and imposes structure on the model itself rather than on the latent space. Finally, the data used in this work come in the form of standard feature vectors as in a typical supervised learning context, rather than as multi-dimensional arrays. Furthermore, [10] considers Bayesian priors on a TN and they employ the Laplace approximation on the posterior, while [11] introduces a TN based method to efficiently capture the conditional probabilities of multiple sets of events with polynomial complexity. Overall, to the best of our knowledge, this is the first work to consider stochastic variational inference in the context of TN methods for supervised machine learning, as well as the first work to consider structured posterior distributions that exploit the multi-linear nature of TNs, making it possible to account for meaningful model interpretations.

Paper Organization: We first present the TN preliminaries necessary to follow this work in Section 2, followed by Mean-Field and Structured BTNs in Section 3, along with their corresponding learning and inference algorithms. Finally, experimental results are given in Section 4, with the conclusions, model limitations, and future research directions in Sections 5 and 6.

2 Preliminaries

2.1 Tensors and Tensor Networks

A real-valued tensor is a multidimensional array, denoted by a calligraphic font, e.g., $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$, where P is the order of the tensor, and I_p ($1 \leq p \leq P$) the size of its p^{th} mode. Matrices (denoted by bold capital letters, e.g., $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$) can be seen as order-2 tensors ($P = 2$), vectors are denoted by bold lower-case letters, e.g., $\mathbf{x} \in \mathbb{R}^I$ and can be seen as order-1 tensors ($P = 1$), and scalars (denoted by lower-case letters, e.g., $x \in \mathbb{R}$) are tensors of order $P = 0$. A specific entry of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ is given by $x_{i_1,\ldots,i_P} \in \mathbb{R}$.

The following conventions for basic linear/multilinear operations are employed throughout the paper. The *outer product* of two vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$ is given by $\mathbf{c} = \mathbf{a} \circ \mathbf{b} \in \mathbb{R}^{I \times J}$, with $c_{i,j} = a_i b_j$. The *Kronecker product* of two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{IK \times JL}$, with $c_{(i-1)K+k,(j-1)L+l} = a_{i,j} b_{k,l}$. The *Hadamard product* of two order-P tensors, $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ and $\mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ is denoted by $\mathcal{C} = \mathcal{A} \circledast \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$, with $c_{i_1,\ldots,i_P} = a_{i_1,\ldots,i_P} b_{i_1,\ldots,i_P}$. Finally, the *inner product* of two order-P tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ and $\mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ is denoted by $c = \langle \mathcal{A}, \mathcal{B} \rangle \in \mathbb{R}$ with $c = \sum_{i_1,\ldots,i_P} a_{i_1,\ldots,i_P} b_{i_1,\ldots,i_P}$.

A Tensor Network (TN) is a tensor architecture comprised of smaller-order core tensors which are connected by tensor contractions, where each tensor is represented as a node, while the number of edges that extend from a given node corresponds to its tensor order. Special instances of tensor networks include those based on Tensor Decomposition (TD) methods, which approximate high-order, large-dimension tensors via contractions of smaller core tensors, therefore drastically reducing the computational complexity in tensor manipulation while preserving the data structure [12].

Canonical Polyadic Decomposition (CPD) [13, 14] is a Tensor Decomposition which expresses a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ as a sum of outer products of vectors $\mathbf{a}_r^{(1)}, \mathbf{a}_r^{(2)}, \ldots, \mathbf{a}_r^{(P)}$ (i.e., rank-1 terms):

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(P)} = \sum_{r=1}^{R} \bigcirc_{p=1}^{P} \mathbf{a}_r^{(p)}, \tag{1}$$

where R is the CP rank of the tensor (equal to the standard matrix rank when $P = 2$). We can arrange these vectors into P factor matrices $\mathbf{A}^{(p)} = [\mathbf{a}_1^{(p)}, \ldots, \mathbf{a}_R^{(p)}] \in \mathbb{R}^{I_n \times R}$, $(1 \leq p \leq P)$, which allows the CPD to be expressed as a contraction of the identity tensor, $\mathcal{I} \in \mathbb{R}^{I_1 \times \cdots \times I_P}$ (super-diagonal values being all one) with the so-formed factor matrices. This view allows for the CPD to be formulated as a TN.

2.2 Tensor Networks for Supervised Learning

Consider a supervised learning task whereby each sample is represented through a set of P features. A *local feature mapping* $\phi \colon \mathbb{R} \to \mathbb{R}^d$ is next applied to every feature, x_p, with d as the *local dimension* of the mapping. The choice of the feature map, ϕ, is flexible and application-dependent. The outer product between the mapped feature vectors then yields

$$\Phi(\mathbf{x}) = \bigcirc_{p=1}^{P} \phi(x_p) \in \mathbb{R}^{d^P}, \tag{2}$$

where $\mathbb{R}^{d^P}$ denotes $\mathbb{R}^{\overbrace{d \times \cdots \times d}^{P \text{ times}}}$. For one output (e.g., single-target regression or binary classification), the prediction in (2) is given by

$$g(\mathbf{x}) = \langle \Phi(\mathbf{x}), \mathcal{W} \rangle, \tag{3}$$

where $\mathcal{W} \in \mathbb{R}^{d^P}$ is the *weight tensor*, which comprises all model coefficients.

Remark 1: The size of the weight tensor, $\mathcal{W}$, scales exponentially with the number of features and is therefore computationally prohibitive to learn. To this end, $\mathcal{W}$ can be represented as a TN [15], to make the number of parameters scale linearly with the number of features.

Depending on the nature of the task at hand, different TNs can be employed for the representation of the weight tensor. In contrast to neural networks which comprise consecutive non-linear layers, the only non-linearity in the proposed model arises from the non-linear feature maps, while all other operations represent linear contractions. This way, it is possible to construct highly expressive, bottom-up models with enhanced interpretability, starting from a well-defined feature mapping ϕ.

3 Variational Bayesian Tensor Networks

Training a deterministic tensor network amounts to learning appropriate values for the entries in the TN-decomposed parameter tensor, $\mathcal{W}$. In order to extend this framework to the Bayesian setting, we treat model parameters as random variables, so that the training goal then becomes that of calculating the posterior distribution of the weights given the training data, $p(\mathcal{W}|D)$. The posterior distribution can be subsequently used for predictive purposes on unseen data, by averaging over the predictions of each possible configuration of the model parameters, and weighted according to the learnt posterior distribution. As is also the case with neural networks, the true posterior distribution is intractable; we therefore resort to approximations, namely variational inference.

In variational learning, the true posterior distribution, $p(\mathcal{W}|D)$, is approximated by a learnable tractable variational posterior, $q_{\boldsymbol{\theta}}(\mathcal{W}|D)$. In the sequel, conditioning on D is removed from the notation for notational simplicity. Variational inference aims to learn the parameters $\boldsymbol{\theta}$ that parameterize the weight distribution, $q_{\boldsymbol{\theta}}(\mathcal{W})$, by minimizing the Kullback-Leibler (KL) divergence between the true and variational posterior distributions. Following standard variational learning methodologies, it is straightforward to show that

$$\begin{aligned} \boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, KL\left[q_{\boldsymbol{\theta}}(\mathcal{W})||p(\mathcal{W}|D)\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, KL\left[q_{\boldsymbol{\theta}}(\mathcal{W})||p(\mathcal{W})\right] - \mathbb{E}_{q_{\boldsymbol{\theta}}(\mathcal{W})}[\log p(D|\mathcal{W})] \end{aligned} \tag{4}$$

Stochastic gradient descent can be used to minimize this cost. Denote the argument of the argmin function in equation (4) by $\mathcal{F}(D, \boldsymbol{\theta})$. Then, it can be approximated [16] through Monte-Carlo sampling as

$$\mathcal{F}(D, \boldsymbol{\theta}) \approx \sum_{i=1}^{N} \log q_{\boldsymbol{\theta}}(\mathcal{W}^{(i)}) - \log p(\mathcal{W}^{(i)}) - \log p(D|\mathcal{W}^{(i)}) \tag{5}$$

where $\mathcal{W}^{(i)}$ is a sample from the variational posterior, $q_{\boldsymbol{\theta}}(\mathcal{W})$. The first term in (4) can be computed either in a closed form as in the case of a tractable prior and variational posterior distributions, or approximated through sampling, as a mean for gradient variance reduction.

3.1 The Mean-Field Approach

In the common mean-field approach, the weight posterior is approximated by a diagonal multivariate Gaussian distribution, leading to mean-field Bayesian tensor networks (MF-BTN), whereby each weight is treated as a separate univariate distribution and the learning procedure consists of learning a separate mean and variance.

We omit the details of the training and inference algorithms since the procedure is outlined in [16]. This approach leads to a higher (double) parameter complexity compared to the deterministic approach, since a separate mean and variance are learnt for every weight. No difference in performance was found between an approximation through sampling and using the closed form KL divergence between the prior and variational posterior distributions in this case.

Even though this approximation allows for a tractable and fast inference over the weight distributions, the independence assumption over the weights proves to be too restrictive [8]. To this end, a structured approach that embarks upon the structural properties of TNs to enhance posterior flexibility is introduced in the next section.

3.2 Tensor-Variate Structured Posterior

In the following, we provide the derivation for the case of CPD, used in our regression experiments. The proposed method leads to Structured Posterior Bayesian Tensor Networks (SP-BTN).

Tensor Gaussian Distribution A random tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, exhibits a Gaussian distribution, defined by the tensor mean, $\mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and mode-n covariance, $\mathbf{R}^{(n)} \in \mathbb{R}^{I_n \times I_n}$, if and only if its vector representation, $\mathbf{x} = vec(\mathcal{X}) \in \mathbb{R}^K$, where $K = \prod_{i=1}^{N} I_i$, is distributed according to [17]

$$\mathbf{x} \sim \mathcal{N}\left(\mathbf{m}, \bigotimes_{i=N}^{1} \mathbf{R}^{(i)}\right) \tag{6}$$

where $\mathbf{m} = vec(\mathcal{M})$.

Reinterpretation of the Weight Tensor Note that the learnable weights of a CPD-decomposed weight tensor $\mathcal{W}$ form a 3rd order tensor. Indeed, we have P factor matrices, each with dimensionality $d \times R$, as defined[1] in Section 2. Instead of considering every weight as being drawn from an independent univariate Gaussian distribution, the weight structure of the CPD admits the following reinterpretation. The parameters of a CPD-decomposed weight tensor, $\mathcal{W}$, can be considered as a single realization of a tensor-variate Gaussian distribution with tensor mean, $\mathcal{M} \in \mathbb{R}^{d \times R \times P}$, and mode-$n$ covariance, $\mathbf{R}^{(n)} \in \mathbb{R}^{n \times n}$, where $n \in \{d, R, P\}$.

When adapting the weight tensor reinterpretation of the variational learning framework, it can be assumed that the variational posterior of the weight tensor is a tensor-variate Gaussian distribution. Note that the parameters of the tensor-variate Gaussian cannot simply be estimated by e.g., maximum likelihood estimation, as this is only possible with tensor-valued observations. In contrast, we here

[1] Note that in the case of CPD, even though $\mathcal{W}$ is originally a P-th order tensor, this is achieved through the contraction of the factor matrices with the superdiagonal tensor as explained in Section 2. In this case, we can work with P factor matrices, each $d \times R$, which yields a 3rd order tensor

deal with lower dimensional typical feature vectors and labels. The probability density function of the tensor valued variational posterior is thus defined as

$$q_{\boldsymbol{\theta}}(\mathcal{W}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{w}-\mathbf{m})^T(\bigotimes_{i\in\{d,R,P\}}\mathbf{R}^{(i)})^{-1}(\mathbf{w}-\mathbf{m})\right]}{(2\pi)^{\frac{dRP}{2}}\det{}^{\frac{1}{2}}\left[\bigotimes_{i\in\{d,R,P\}}\mathbf{R}^{(i)}\right]} \tag{7}$$

where $\boldsymbol{\theta} = \{\mathbf{m}, \mathbf{R}^{(i)}\}$ denotes the trainable variational parameters.

Remark 2: A Kronecker structure on the mode-n covariances is imposed by the definition of the tensor-variate Gaussian distribution. For computational tractability, we consider diagonal mode-n covariances, so that the resulting global covariance matrix $\bigotimes_{i\in\{d,R,P\}}\mathbf{R}^{(i)}$ is also diagonal which mitigates the computational bottleneck of inverting a dense covariance matrix. Futhermore, $diag\left(\bigotimes_{i\in\{d,R,P\}}\mathbf{R}^{(i)}\right) = \bigotimes_{i\in\{d,R,P\}} diag\left(\mathbf{R}^{(i)}\right)$, which allows us to employ only the diagonal part of the mode-n covariance matrices, thus further saving on the memory and time resources. For clarity, we shall use the notation $diag\left(\mathbf{R}^{(d)}\right) = \left(w_1^d, w_2^d, \ldots, w_d^d\right)^T$, $diag\left(\mathbf{R}^{(R)}\right) = \left(w_1^R, w_2^R, \ldots, w_R^R\right)^T$, and $diag\left(\mathbf{R}^{(P)}\right) = \left(w_1^P, w_2^P, \ldots, w_P^P\right)^T$. Since $\mathbf{a}\otimes\mathbf{b} = vec(\mathbf{a}\circ\mathbf{b})$, it follows that $\bigotimes_{i\in\{d,R,P\}}\mathbf{R}^{(i)} = \left(w_1^d w_1^R w_1^P, w_1^d w_1^R w_2^P, \ldots, w_d^d w_R^R w_P^P\right)$.

Remark 3: Assuming a separate variance for every weight amounts to having dRP parameters, while the proposed assumption of Kronecker separability yields a linear scaling with $(d+R+P)$ parameters. By virtue of Kronecker separability, the proposed approach therefore introduces inter-weight information sharing at a greatly reduced parameter complexity.

Next, instead of considering a separate mean for every weight distribution, we impose a low rank structure on the tensor mean, $\mathcal{M}$, in the form

$$\mathcal{M} = \sum_{m=1}^{M_R}\mathbf{b}_m^{(d)}\circ\mathbf{b}_m^{(R)}\circ\mathbf{b}_m^{(P)} \tag{8}$$

where M_R denotes mean rank [7].

This approximation further reduces parameter complexity while preserving the expressiveness of the posterior by introducing weight dependencies. More specifically, the proposed approximation amounts to $(d+R+P)M_R$ parameters, in comparison to the dRP parameters in the mean-field case. In practice, it was found that very low ranks were too restrictive, but after a relatively low number (e.g, $M_R = 10$), the posterior mean was flexible enough for efficient model learning.

Prior Distribution We have chosen the multivariate diagonal Gaussian distribution given by

$$p(\mathcal{W}) = \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_{dRP}) \tag{9}$$

as the prior distribution. In this work, we diverge from the approach in [16], where a scale mixture of two Gaussian densities was suggested in order to a-priori cluster many weights around zero. Also, TNs can become sensitive to initialization, especially for large P, due to the many matrix (for CPD) or tensor (for the tensor train) multiplications, as has been previously pointed out in [1, 5]. In other words, the final result can vanish or explode if every initialized matrix/tensor is too small or too big. In practice we found that, in general, a normal distribution with standard deviation equal to 0.5 worked well for our experiments, but the issue was not completely eliminated (see Section 5).

Tensor Posterior Sampling For efficient stochastic variational inference, it is necessary to extend the reparameterization trick [16] for the tensor-variate posterior distribution. In this case, sampling from the posterior amounts to sampling random tensor noise, $\mathcal{E}\in\mathbb{R}^{d\times R\times P}$, from a standard Gaussian distribution. A sample from the posterior is then obtained as $\mathcal{W}^{(i)} = \mathcal{M} + \mathcal{E}\times L_d\times L_R\times L_P$, where L_i denotes the Cholesky factor of the corresponding mode-n covariance matrix $\mathbf{R}^{(i)}, i\in\{d,R,P\}$, and $\times$ denotes the mode-n product.

KL-Divergence between Tensor-Variate Gaussians It was empirically found that training stability is significantly enhanced when using the closed-form KL divergence. To explicitly obtain the KL-divergence between the prior and the variational posterior, we employ Equation (6) and the

KL-divergence between two multivariate Gaussians, to yield

$$
\begin{aligned}
&KL\left[q_{\boldsymbol{\theta}}(\mathcal{W})||p(\mathcal{W})\right] = \\
&= \frac{1}{2}[\mathrm{tr}\left(\left(\bigotimes_{i\in\{d,R,P\}} \mathbf{R}_p^{(i)}\right)^{-1} \bigotimes_{i\in\{d,R,P\}} \mathbf{R}_q^{(i)}\right) \\
&+ \left(\mathbf{m}_p - \mathbf{m}_q\right)^T \left(\bigotimes_{i\in\{d,R,P\}} \mathbf{R}_p^{(i)}\right)^{-1} \left(\mathbf{m}_p - \mathbf{m}_q\right) \\
&- dRP + \log \frac{\det \bigotimes_{i\in\{d,R,P\}} \mathbf{R}_p^{(i)}}{\det \bigotimes_{i\in\{d,R,P\}} \mathbf{R}_q^{(i)}}]
\end{aligned} \tag{10}
$$

The cost in (4) can now be minimized using gradient descent, with the expression in (10) for the closed form KL-divergence and the proposed posterior given in (7).

Remark 4: The proposed probabilistic approach to learn the model parameters considers the weight tensor as a realization of a tensor-variate Gaussian distribution. This makes it possible to induce information sharing between weights, and at a significantly lower parameter complexity compared to the mean-field case, while the flexibility of the tensor mean is controlled by the mean rank hyperparameter.

4 Experiments

A polynomial mapping of features with unit norm was employed,

$$
\hat{\phi}_d(x_n) = \frac{1}{\sqrt{\sum_{k=0}^{d-1} x_n^{2k}}} \left[1, x_n, x_n^2, \ldots, x_n^{(d-1)}\right]^T \tag{11}
$$

as suggested in [1]. This enhances training stability when a high local dimension is employed. Note that no unit norm was used for the toy dataset since the ground truth local dimension was low ($d = 4$).

4.1 Mean Field vs Structured Posterior BTNs

As this is the first work to consider variational BTNs, we start with a brief "proof of concept" experiment to establish if the so induced flexibility in the posterior distribution can efficiently enhance model expressiveness at a reduced parameter complexity. To this end, we employed the standard regression California Housing Dataset. We used the complete dataset for training, as our goal in this experiment was to establish the expressiveness of the models based on their ability to fit an outlier-rich and medium size dataset (20640 samples). Features and labels were standardized to zero mean and unit variance, the mini-batch size was set to 128, and the Adam optimizer [18] with a learning rate of 0.002 was used in training. Both models were trained over 1000 epochs to ensure convergence.

The CPD rank was set to $R = 20$, and the local dimension to $d = 75$. The dataset contained $P = 8$ features with a total of $2dRP = 24000$ parameters for the mean-field case. Regarding the structured posterior, the model contained $(d + R + P)(M_R + 1)$ parameters, a total of $103M_R$.

Figure 1 shows the training MSE in terms of the number of model parameters. Three important conclusions can be made from this figure. First, observe that the training error of the SP-BTN drops below that of MF-BTN after $M_R = 70$ (corresponding to 7210 parameters), illustrating the desired gains in expressiveness from the structured posterior at a significantly lower model complexity (7210 vs 24000 parameters). Secondly, after $M_R = 100$, the training error reaches a plateau due to the information bottleneck imposed by the low *CPD* rank. Finally, for a very low mean rank, the model underfits since the coupling between weight means is too strong and inhibits efficient learning.

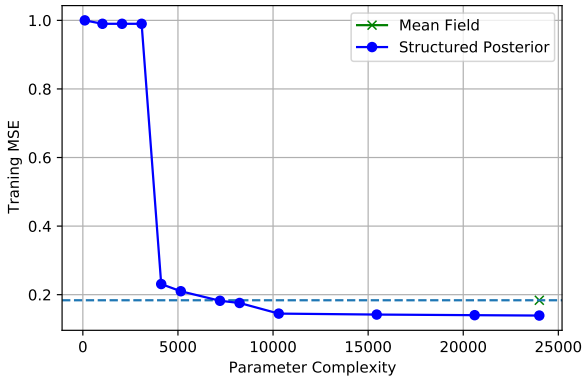

Figure 1: Training MSE as a function of model parameter complexity

4.2 Performance Evaluation over a Synthetic Dataset

The ability of the proposed Bayesian TN to provide uncertainty quantification was assessed over an experiment based on toy data. The proposed MF- and SP-BTN models were examined on a simple toy dataset that was used in [19], whereby 20 inputs were sampled from $\mathcal{U}[-4, 4]$ so as to construct the target variable, $y_n = x_n^3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 9)$. Two low complexity models were examined at data fitting, a MF-BTN network with $R = 5$ and 80 parameters overall, and a SP-BTN with $R = 2$, $M_R = 10$, and with 77 parameters overall. Figure 2 shows the corresponding predictive distributions.

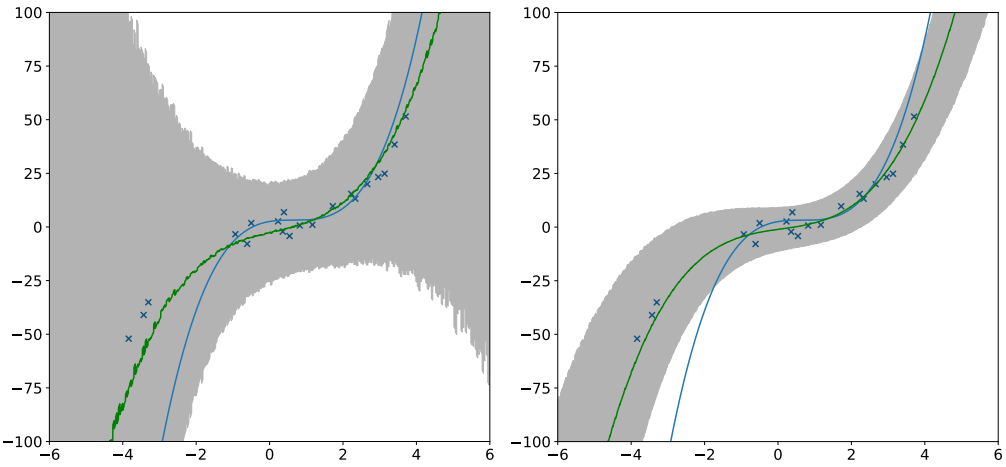

Figure 2: Predictive distributions for the toy dataset for MF-BTN (left) and SP-BTN (right). The crosses represent the training points, while blue curves correspond to the true function, and green curves to predictive mean functions. The grey areas designate the ± 3 standard deviations around the mean.

Observe that the MF-BTN underfitted due to the limited amount of datapoints, leading to high predictive uncertainty. However, the MF-BTN was able to (approximately) learn the mean function, in contrast to the corresponding BNN with factorized Gaussian [8]. This is likely due to the fact that in this case the BTNs have an advantage due to the polynomial nature of both the true function and the feature mapping.

Regarding the SP-BTN, it provided a realistic predictive distribution; for a visual comparison with common BNN models, we refer the reader to [8]. It should be mentioned that due to the low number of datapoints, the SP-BTN produced a rather high uncertainty in low data density areas, which was

higher than that of the corresponding BNNs, a result of the low rank regularization of TNs that prevents overfitting [1].

Finally, inherent advantages of TNs, such as their interpretability stemming from their multi-linear nature, can be demonstrated by the coefficient computation for each (transformed) feature interaction through

$$\sum_{i_1,\ldots,i_P} \left(\circledast_{k=1}^{P} \mathbf{w}_{i_k}^{(k)} \right) \tag{12}$$

where $\mathbf{w}_{i_k}^{(k)}$ denotes the weight in the P-th factor matrix and i_k-th row, as described in [1]. By virtue of the stochastic nature of the presented framework, it is now possible to compute credibility intervals for individual coefficients. In this example, after model training, the estimated single regression coefficient taking part in the model employed was 0.97 ± 0.05, which includes the ground truth value (equal to 1).

4.3 Benchmark Regression Datasets

The next set of experiments employed benchmark UCI regression datasets [19]. The scores were reported only for SP-BTN, as they were consistently higher than MF-BTN at a comparable parameter complexity. The proposed SP-BTN was evaluated against the Bayes By Backprop (BBB) [16], Variational Matrix Gaussian (VMG) [8], and functional BNN (fBNN) [20] models. Since regression on non-sequential tasks was considered, the CPD was used for the representation of $\mathcal{W}$ [1]. The BTN models were trained until convergence with the Adam optimizer [18], at a learning rate of 0.001, and a mini-batch size of 128. The simulations were conducted on a 2.9 GHz Intel Core i5 with an 8GB RAM, with the runtime ranging between 0.2 and 0.8 seconds per epoch.

4.3.1 Small Scale Datasets

For a fair comparison between BTNs and BNNs, the BTN models were kept simple, with $R = 10$ and $d = 10$, while M_R was ranging between 10 and 30, with higher values used in datasets where $M_R = 10$ turned out to be too restrictive (and thus underfit in the training set). These settings led to parameter complexity similar to that of the BNNs considered (one hidden layer with 50 units, as reported in [20]).

Each dataset was randomly split into the training and test sets, comprising respectively 90% and 10% of the data, and the procedure was repeated 20 times. Table 1 shows the mean and standard deviations for the models considered. Data were normalized so that the input features and the targets had zero mean and unit variance in the training set. The normalization on the targets was inversed during prediction.

Table 1: Averaged test RMSE for small scale regression benchmarks.

Data	BBB	VMG	fBNN	SP-BTN
Wine	0.643 ± 0.012	$\mathbf{0.63 \pm 0.01}$	0.673 ± 0.014	0.6417 ± 0.011
Concrete	5.678 ± 0.08	$\mathbf{4.89 \pm 0.12}$	4.9355 ± 0.18	5.50 ± 0.23
Energy	0.565 ± 0.018	0.54 ± 0.02	$\mathbf{0.412 \pm 0.017}$	0.549 ± 0.2
Yacht	1.174 ± 0.086	0.71 ± 0.05	0.607 ± 0.068	$\mathbf{0.5061 \pm 0.091}$

4.3.2 Large Scale Datasets

For a fair comparison between BTNs and BNNs, values for R, d, and M_R were chosen so as to lead to similar parameter complexity as the BNNs considered (one hidden layer with 100 units, as reported in [20]). The datasets were randomly split into 80% training, 10% validation, and 10% test data, whereby the validation set was employed to tune the hyperparameters[2]. Data were normalized

[2]For Protein and Naval, $R = 25$, $d = 25$ and $M_R = 50$. For GPU, $R = 30$, $d = 30$ and $M_R = 30$. BTN parameter complexity was similar with that of the compared BNNs

so that the input features and the targets had zero mean and unit variance in the training set. The normalization on the targets was inversed during prediction. The experiments were repeated 5 times, with the mean and standard deviation for different models shown Table 3.

Table 2: Averaged test RMSE for large scale regression benchmarks.

Data	BBB	fBNN	SP-BTN
Protein	4.331 ± 0.033	4.326 ± 0.019	$\mathbf{4.25 \pm 0.012}$
Naval	0.01 ± 0.0	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$
GPU	21.886 ± 0.673	19.50 ± 0.171	$\mathbf{19.36 \pm 0.212}$

Overall, it can be observed from Table 2 that the proposed BTN models exhibited comparable performance to state-of-the-art BNNs, at a similar parameter complexity but with enhanced physical intuition.

5 Limitations

It was empirically found that optimization of the TN models can run into numerical difficulties due to the many multiplications involved, as also noted in [5] and [1], this issue is emphasized when many features are present in the dataset at hand (more Hadamard products in our case). Indeed, this issue did cause the convergence to be slow and rendered the optimized parameters sensitive to initialization, and consequently a convergence to suboptimal values in some runs; such runs were excluded from the results. An important future research direction thus concerns the optimization of the proposed models. To this end, natural gradient optimization appears a promising direction that has the potential to guide the optimization towards faster convergence. The low parameter complexity of the proposed models makes this optimization scheme practical and will be an important part of our future work.

6 Conclusion

We have introduced variational inference for tensor networks, a novel framework for employing compressed tensor network models in the Bayesian setting. After presenting a basic mean-field approach, a structured model has been introduced that takes advantage of the multi-linear nature of TNs to efficiently induce information sharing in the posterior distribution of the model weights. We have shown that the so introduced structured Bayesian tensor networks not only yield competitive results compared to popular Bayesian neural network models in benchmark regression datasets, but also that the proposed framework exhibits enhanced physical interpretability. While this work has focused on the regression paradigm, the BTN framework can be readily extended to other domains, such as computer vision, through the appropriate modification of the tensor network used to decompose the weight tensor (e.g., 2-D tensor networks for images).

Broader Impact

An immediate impact of our work is its potential to learn expressive models with enhanced interpretability and the ability for uncertainty quantification. The broader impact, as well as the impact arising from model failure and data bias is identical to those of supervised learning models in general. We have not identified anyone that can be put at disadvantage from this work.

Acknowledgments and Disclosure of Funding

The authors would like to thank Dr. Bruno Scalzo Dees for helpful discussions. K.K. is supported by an EPSRC International Doctoral Scholarship. Y.L.X. is supported by an EPSRC Doctoral Scholarship.

A Large Scale Regression with Deeper Networks

In this appendix we experimented on large scale regression datasets with deeper networks. We compare SP-BTN with BBB and fBNNs, with similar parameter complexity (5 hidden layers of 100 units, as described in [20]. In this case, we used $R = 1000$, $d = 50$ and $M_R = 50$, and maintained the same experimental setup as in section 4.3.2. Also in this setup, we observe that SP-BTNs produce competitive results.

Table 3: Averaged test RMSE for large scale regression benchmarks using deeper networks.

Data	BBB	fBNN	SP-BTN
Protein	3.684 ± 0.041	$\mathbf{3.659 \pm 0.026}$	3.94 ± 0.03
Naval	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$
GPU	5.136 ± 0.087	4.806 ± 0.116	$\mathbf{4.79 \pm 0.135}$

References

[1] A. Haliassos, K. Konstantinidis, and D. P. Mandic. Supervised learning for non-sequential data: A Canonical Polyadic Decomposition Approach. *IEEE Transactions on Neural Networks and Learning Systems*. in print, 2021.

[2] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and A. H. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, March 2015.

[3] A. Cichocki, N. Lee, I. Oseledets, A. Phan, Q. Zhao, and D. P. Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

[4] C. Guo, Z. Jie, W. Lu, and D. Poletti. Matrix product operators for sequence-to-sequence learning. *Physical Review E*, 98:042114, Oct 2018. doi: 10.1103/PhysRevE.98.042114.

[5] J. Wang, C. Roberts, G. Vidal, and S. Leichenauer. Anomaly detection with tensor networks. In *Proceedings of the First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[6] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *Proceedings of the Conference on Learning Theory*, pages 698–728, 2016.

[7] B. Scalzo Dees, A.H. Phan, and D.P. Mandic. A statistically identifiable model for tensor-valued Gaussian random variables *arXiv* 1911.02915, 2019.

[8] C. Louizos and M. Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1708–1716. PMLR, 2016.

[9] A. Campbell and P. Liò. TVGP-VAE: Tensor-variate Gaussian process prior variational autoencoder, *arXiv* 2006.04788, 2020.

[10] Erdong Guo and David Draper. The bayesian method of tensor networks, arXiv 2101.00245, 2021.

[11] Shi-Ju Ran. Bayesian tensor network with polynomial complexity for probabilistic machine learning, arXiv 1912.12923, 2020.

[12] A. Cichocki. Era of big data processing: A new approach via tensor networks and tensor decompositions. In *Proceedings of the International Workshop on Smart Info-Media Systems in Asia*, March 2014.

[13] J.D. Carroll and J.J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. *Psychometrika*, 35(3):283–319, 1970.

[14] R. A. Harshman. *Foundations of the PARAFAC Procedure: Models and conditions for an "explanatory" multi-modal factor analysis*. UCLA working papers in phonetics. University of California at Los Angeles, 1970.

[15] E. Stoudenmire and D. Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems 29*, pages 4799–4807, 2016.

[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1613–1622, 2015.

[17] Martin Singull, M. Ahmad, and Dietrich von Rosen. The multilinear normal distribution: Introduction and some basic properties. *Journal of Multivariate Analysis*, 113, 2011. doi: 10.1016/j.jmva.2011.05.015.

[18] D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[19] J.M. Hernandez-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1861–1869, 2015.

[20] S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional variational Bayesian neural networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]