Quantized Fourier and Polynomial Features for more Expressive Tensor Network Models

Frederiek Wesel
Delft Center for Systems and Control
Delft University of Technology

Kim Batselier
Delft Center for Systems and Control
Delft University of Technology

Abstract

In the context of kernel machines, polynomial and Fourier features are commonly used to provide a nonlinear extension to linear models by mapping the data to a higher-dimensional space. Unless one considers the dual formulation of the learning problem, which renders exact large-scale learning unfeasible, the exponential increase of model parameters in the dimensionality of the data caused by their tensor-product structure prohibits to tackle high-dimensional problems. One of the possible approaches to circumvent this exponential scaling is to exploit the tensor structure present in the features by constraining the model weights to be an underparametrized tensor network. In this paper we quantize, i.e. further tensorize, polynomial and Fourier features. Based on this feature quantization we propose to quantize the associated model weights, yielding quantized models. We show that, for the same number of model parameters, the resulting quantized models have a higher bound on the VC-dimension as opposed to their non-quantized counterparts, at no additional computational cost while learning from identical features. We verify experimentally how this additional tensorization regularizes the learning problem by prioritizing the most salient features in the data and how it provides models with increased generalization capabilities. We finally benchmark our approach on large regression task, achieving state-of-the-art results on a laptop computer.

1 INTRODUCTION

In the context of supervised learning, the goal is to estimate a function $f(\cdot) : \mathcal{X} \to \mathcal{Y}$ given N input-output pairs $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$, where $\boldsymbol{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. Kernel machines accomplish this by lifting the input data into a high-dimensional feature space by means of a *feature map* $\boldsymbol{z}(\cdot) : \mathcal{X} \to \mathcal{H}$ and seeking a linear relationship therein:

$$f(\boldsymbol{x}) = \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{w} \rangle . \tag{1}$$

Training such a model involves the minimization of the regularized empirical risk given a convex measure of loss $\ell(\cdot,\cdot) : \mathcal{H} \times \mathcal{Y} \to \mathbb{R}_+$

$$R_{\text{empirical}}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \ell\left(\langle \boldsymbol{z}(\boldsymbol{x}_n), \boldsymbol{w} \rangle, y_n\right) + \lambda \left|\left|\boldsymbol{w}\right|\right|^2 . \tag{2}$$

Different choices of loss yield the *primal* formulation of different kernel machines. For example, squared loss results in kernel ridge regression (KRR) (Suykens et al., 2002), hinge loss in support vector machines (SVMs) (Cortes and Vapnik, 1995), and logistic loss yields logistic regression. Different choices of the feature map $\boldsymbol{z}$ allow for modeling different nonlinear behaviors in the data. In this article we consider tensor-product features

$$\boldsymbol{z}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}(x_d) , \tag{3}$$

where $\boldsymbol{v}^{(d)}(\cdot) : \mathbb{C} \to \mathbb{C}^{M_d}$ is a feature map acting on each element of the d-th component x_d of $\boldsymbol{x} \in \mathbb{C}^D$. Here $\otimes$ denotes the *left* Kronecker product (Cichocki et al., 2016). This tensor-product structure arises when considering product kernels (Shawe-Taylor and Cristianini, 2004; Hensman et al., 2017; Solin and Särkkä, 2020), Fourier features (Wahls et al., 2014), when considering B-splines (Karagoz and Batselier, 2020) and polynomials (Shawe-Taylor and Cristianini, 2004).

Due to the tensor-product structure in equation (3), $\boldsymbol{z}(\cdot)$ maps an input sample $\boldsymbol{x} \in \mathbb{C}^D$ into an exponentially large feature vector $\boldsymbol{z}(\boldsymbol{x}) \in \mathbb{C}^{M_1 M_2 \cdots M_D}$. As a result,

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

the model is also described by an exponential number of weights $\boldsymbol{w}$. This exponential scaling in the number of features limits the use of tensor-product features to low-dimensional data or to mappings of very low degree.

Both these computational limitations can be sidestepped entirely by considering the *dual* formulation of the learning problem in equation (2), requiring to compute the pairwise similarity of all data respectively by means of a kernel function $k(\boldsymbol{x}, \boldsymbol{x'}) = \langle \boldsymbol{z}(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{x'}) \rangle$. However, the dual formulation requires to instantiate the kernel matrix at a cost of $\mathcal{O}(N^2)$ and to estimate N Lagrange multipliers by solving a (convex) quadratic problem at a cost of at least $\mathcal{O}(N^2)$, prohibiting to tackle large-scale data (large N). To lift these limitations, a multitude of research has focused on finding low-rank approximations of kernels by considering *random* methods such as polynomial sketching (Pham and Pagh, 2013; Woodruff, 2014; Meister et al., 2019) and random features (Williams and Seeger, 2001; Rahimi and Recht, 2007; Le et al., 2013), which approximate the feature space with probabilistic approximation guarantees.

One way to take advantage of the existing tensor-product structure in equation (3) is by imposing a tensor network (Kolda and Bader, 2009; Sidiropoulos et al., 2017) constraint on the weights $\boldsymbol{w}$. For example, using a polyadic rank-R constraint reduces the storage complexity of the weights from $\mathcal{O}(M^D)$ down to $\mathcal{O}(DMR)$ and enables the development of efficient learning algorithms with a computational complexity of $\mathcal{O}(DMR)$ per gradient descent iteration. This idea has been explored for polynomial (Favier and Bouilloc, 2009; Rendle, 2010; Blondel et al., 2016, 2017; Batselier et al., 2017) pure-power-1 polynomials (Novikov et al., 2018), pure-power polynomials of higher degree (Chen et al., 2018), B-splines (Karagoz and Batselier, 2020), and Fourier features (Wahls et al., 2014; Stoudenmire and Schwab, 2016; Efthymiou et al., 2019; Kargas and Sidiropoulos, 2021; Cheng et al., 2021; Wesel and Batselier, 2021).

In this article, we improve on this entire line of research by deriving an exact *quantized* representation (Khoromskij, 2011) of pure-power polynomials and Fourier features, exploiting their inherent Vandermonde structure. It is worth noting that *in this paper quantized means further tensorized*, and should not be confused with the practice of working with lower precision floating point numbers. By virtue of the derived quantized features, we are able to quantize the model weights. We show that compared to their non-quantized counterparts, quantized models can be trained with no additional computational cost, while learning from the same exact features. Most importantly, for the same number of model parameters the ensuing quantized models are characterized by higher upper bounds on the VC-dimension, which indicates a potential higher expressiveness. While these bounds are in practice not necessarily met, we verify experimentally that:

1. Quantized models are indeed characterized by higher expressiveness. This is demonstrated in section 5.1, where we show that in the under-parameterized regime quantized models achiever lower test errors than the non-quantized models with identical features and identical total number of model parameters.
2. This additional structure regularizes the problem by prioritizing the learning of the peaks in the frequency spectrum of the signal (in the case of Fourier features) (section 5.2). In other words, the quantized structure is learning the most salient features in the data first with its limited amount of available model parameters.
3. Quantized tensor network models can provide state-of-the-art performance on large-scale real-life problems. This is demonstrated in section 5.3, where we compare the proposed quantized model to both its non-quantized counterpart and other state-of-the-art methods, demonstrating superior generalization performance on a laptop computer.

2 BACKGROUND

We denote scalars in both capital and non-capital italics w, W, vectors in non-capital bold $\boldsymbol{w}$, matrices in capital bold $\boldsymbol{W}$ and tensors, also known as higher-order arrays, in capital italic bold font $\boldsymbol{\mathcal{W}}$. Sets are denoted with calligraphic capital letters, e.g. $\mathcal{S}$. The m-th entry of a vector $\boldsymbol{w} \in \mathbb{C}^M$ is indicated as w_m and the $m_1 m_2 \dots m_D$-th entry of a D-dimensional tensor $\boldsymbol{\mathcal{W}} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ as $w_{m_1 m_2 \dots m_D}$. We denote the complex-conjugate with superscript * and $\otimes$ denotes the *left* Kronecker product (Cichocki et al., 2016). We employ zero-based indexing for all tensors. The Frobenius inner product between two D-dimensional tensors $\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{W}} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ is defined as

$$\langle \boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{W}} \rangle := \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2-1} \cdots \sum_{m_D=0}^{M_D-1} v^*_{m_1 m_2 \dots m_D} w_{m_1 m_2 \dots m_D}. \tag{4}$$

We define the vectorization operator as $\texttt{vec}(\cdot) : \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D} \to \mathbb{C}^{M_1 M_2 \cdots M_D}$ such that

$$\texttt{vec}(\boldsymbol{\mathcal{W}})_m = w_{m_1 m_2 \dots m_D},$$

with $m = m_1 + \sum_{d=2}^{D} m_d \prod_{k=1}^{d-1} M_k$. Likewise, its inverse, the tensorization operator $\texttt{ten}(\cdot, M_1, M_2, \dots, M_D) : \mathbb{C}^{M_1 M_2 \cdots M_D} \to$

$\mathbb{C}^{M_1 \times M_2 \times \ldots M_D}$ is defined such that

$$\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right)_{m_1 m_2 \cdots m_D} = w_m.$$

2.1 Tensor Networks

Tensor networks (TNs) (Kolda and Bader, 2009; Cichocki, 2014; Cichocki et al., 2016, 2017) express a D-dimensional tensor $\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right) =: \boldsymbol{\mathcal{W}}$ as a multi-linear function of C *core* tensors, see definition A.1 for a rigorous definition. Two commonly used TNs are the canonical polyadic decomposition (CPD) and tensor train (TT).

Definition 2.1 (Canonical polyadic decomposition (Hitchcock, 1927; Kolda and Bader, 2009))**.** A D-dimensional tensor $\boldsymbol{\mathcal{W}} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ has a rank-R CPD if

$$w_{m_1 m_2 \ldots m_D} = \sum_{r=0}^{R-1} \prod_{d=1}^{D} {w^{(d)}}_{m_d r}.$$

The cores of this particular network are $C = D$ matrices $\boldsymbol{W}^{(d)} \in \mathbb{C}^{M_d \times R}$. The storage complexity $P = R \sum_{d=1}^{D} M_d$ of a rank-R CPD is therefore $\mathcal{O}(DMR)$, where $M = \max(M_1, M_2, \ldots, M_D)$.

Definition 2.2 (Tensor train (Oseledets, 2011))**.** A D-dimensional tensor $\boldsymbol{\mathcal{W}} \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_D}$ admits a rank-$(R_1 := 1, R_2, \ldots, R_D, R_{D+1} := R_1)$ tensor train if

$$w_{m_1 m_2 \ldots m_D} = \sum_{r_1=0}^{R_1-1} \sum_{r_2=0}^{R_2-1} \cdots \sum_{r_D=0}^{R_d-1} \prod_{d=1}^{D} {w^{(d)}}_{r_d m_d r_{d+1}}.$$

The cores of a tensor train are the $C = D$ 3-dimensional tensors $\boldsymbol{\mathcal{W}}^{(d)} \in \mathbb{C}^{R_d \times M \times R_{d+1}}$. The case $R_1 > 1$ is also called a tensor ring (TR) (Zhao et al., 2016). Throughout the rest of this article we will simply refer to the tensor train rank as $R = \max(R_2, \cdots, R_D)$. The storage complexity $P = \sum_{d=1}^{D} M_d R_d R_{d+1}$ of a tensor train is then $\mathcal{O}(DMR^2)$. A TN is *underparametrized* if $P \ll \prod_{d=1}^{D} M_d$, i.e. it can represent a tensor with fewer parameters than the number of entries of the tensor.

Other TNs are the Tucker decomposition (Tucker, 1963, 1966), hierarchical hierarchical Tucker (Hackbusch and Kühn, 2009; Grasedyck, 2010) decomposition, block-term decompositions (De Lathauwer, 2008a,b), PEPS (Verstraete and Cirac, 2004) and MERA (Evenbly and Vidal, 2009).

2.2 Tensorized Kernel Machines

The tensor-product structure of features in equation (3) can be exploited by imposing a tensor network structure onto the tensorized model weights

$$\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right).$$

Although generally speaking the tensorized model weights are not full rank, modeling them as an underparametrized tensor network allows to compute fast model responses when the feature map $\boldsymbol{z}\left(\cdot\right)$ is of the form of equation (3).

Theorem 2.3 (Tensorized kernel machine (TKM))**.** *Suppose* $\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right)$ *is a tensor in CPD, TT or TR form. Then model responses and associated gradients*

$$f\left(\boldsymbol{x}\right) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}\left(x_d\right), \boldsymbol{w}\rangle,$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = DMR$ *in case of CPD, and* $P = DMR^2$ *in case of TT or TR.*

Proof. See appendix B.1. □

Results for more general TNs can be found in appendix B.1. This idea has been explored for a plethora of different combinations of tensor-product features and tensor networks (Wahls et al., 2014; Stoudenmire and Schwab, 2016; Novikov et al., 2018; Chen et al., 2018; Cheng et al., 2021; Khavari and Rabusseau, 2021; Wesel and Batselier, 2021). A graphical depiction of a TKM can be found in figure 1a: a full line denotes a summation along the corresponding index, while a dotted line denotes a Kronecker product. Training a kernel machine under such constraint yields the following nonconvex optimization problem:

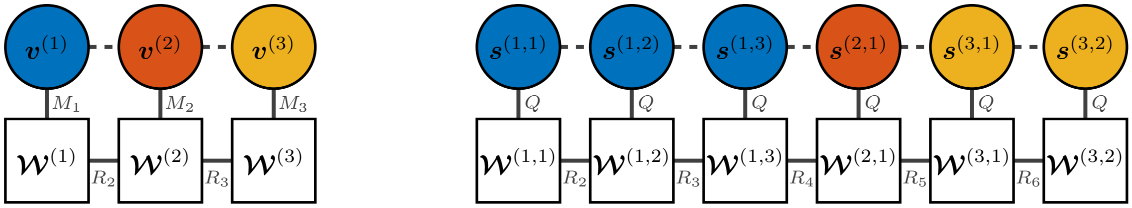
$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}\left(x_d\right), \boldsymbol{w}\rangle, y_n) + \lambda \left|\left|\boldsymbol{w}\right|\right|^2, \quad (5)$$
$$\text{s.t. } \texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right) \text{ is a tensor network.}$$

Common choices of tensor network-specific optimizers are the alternating linear scheme (ALS) (Comon et al., 2009; Kolda and Bader, 2009; Uschmajew, 2012; Holtz et al., 2012), the density matrix renormalization Group (DMRG) (White, 1992) and Riemannian optimization (Novikov et al., 2018, 2021). Generic first or second order gradient-based optimization method can also be employed.

3 QUANTIZING POLYNOMIAL AND FOURIER FEATURES

Before presenting the main contribution of this article, we first provide the definition of a pure-power polynomial feature map.

Definition 3.1 (Pure-power polynomial feature map (Chen et al., 2018))**.** For an input sample $\boldsymbol{x} \in \mathbb{C}^D$, the pure-power polynomial features $\boldsymbol{z}(\cdot) : \mathbb{C}^D \rightarrow$



(a) TKM with TT-constrained weights.

(b) Corresponding QTKM with Q-quantized TT-constrained weights.

Figure 1: TKM (figure 1a), and QTKM (figure 1b) with TT-constrained model weights. In these diagrams, each circle represent a vector which constitutes the pure-power feature map of definition 3.1, and each square represents a tensor train core (definition 2.2). The color coding relates the d-th feature with its quantized representation. A full connecting line denotes a summation along the corresponding index, while a dotted line denotes a Kronecker product, see Cichocki et al. (2016) for a more in-depth explanation. Figure 1b depicts the case where $K_1 = Q^3$, $K_2 = Q$ and $K_3 = Q^2$. Notice how quantization allows to model correlations within each particular mode of the model weights, in this case explicitly by means of the tensor train ranks $(1, R_2, \ldots, R_6, 1)$.

$\mathbb{C}^{M_1 M_2 \cdots M_D}$ of degree $(M_1 - 1, M_2 - 1, \ldots, M_D - 1)$ are defined as

$$\boldsymbol{z}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}(x_d),$$

with $\boldsymbol{v}^{(d)}(\cdot) : \mathbb{C} \to \mathbb{C}^{M_d}$ the Vandermonde vector

$$\boldsymbol{v}^{(d)}(x_d) = \left[1, x_d, x_d^2, \ldots, x_d^{M_d - 1}\right].$$

The m_d-th element of the feature map vector $\boldsymbol{v}^{(d)}(x_d)$ is

$$v^{(d)}(x_d)_{m_d} = (x_d)^{m_d}, \quad m_d = 0, 1, \ldots, M_d - 1.$$

The definition of the feature map is given for degree $(M_1 - 1, M_2 - 1, \ldots, M_D - 1)$ such that the feature map vector $z(\boldsymbol{x})$ has a length $M_1 M_2 \cdots M_D$. The Kronecker product in definition 3.1 ensures that all possible combinations of products of monomial basis functions are computed, up to a total degree of $\sum_{d=1}^{D}(M_d - 1)$. Compared to the more common affine polynomials, which are basis functions of the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x'}) = (b + \langle \boldsymbol{x}, \boldsymbol{x'} \rangle)^M$, pure-power polynomial features contain more higher-order terms. Similarly, their use is justified by the Stone-Weierstrass theorem (De Branges, 1959), which guarantees that any continuous function on a locally compact domain can be approximated arbitrarily well by polynomials of increasing degree. Fourier features can be similarly defined by replacing the monomials with complex exponentials.

Definition 3.2. (Fourier Features) For an input sample $\boldsymbol{x} \in \mathbb{C}^D$, the Fourier feature map $\boldsymbol{\varphi}(\cdot) : \mathbb{C}^D \to \mathbb{C}^{M_1 M_2 \cdots M_D}$ with M_d basis frequencies $-M_d/2, \ldots, M_d/2 - 1$ per dimension is defined as

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \left(c_d \, \boldsymbol{v}^{(d)} \left(e^{-\frac{2\pi j x_d}{L}} \right) \right),$$

where j is the imaginary unit, $c_d = e^{2\pi j x_d \frac{2+M_d}{2L}} \in \mathbb{C}$, $L \in \mathbb{R}$ is the periodicity of the function class and $\boldsymbol{v}^{(d)}(\cdot)$ are the Vandermonde vectors of definition 3.1.

Fourier features are ubiquitous in the field of kernel machines as they are eigenfunctions of D-dimensional stationary product kernels with respect to the Lebesgue measure, see (Rasmussen and Williams, 2006, Chapter 4.3) or (Hensman et al., 2017; Solin and Särkkä, 2020). As such they are often used for the uniform approximation of such kernels in the limit of $L \to \infty$ and $M_1, M_2, \ldots, M_D \to \infty$ (Wahls et al., 2014, Proposition 1).

We now present the first contribution of this article, which is an exact *quantized*, i.e. further tensorized, representation of pure-power polynomials and Fourier features. These quantized features allows for the quantization of the model weights, which enables to impose additional tensor network structure between features, yielding more expressive models for the same number of model parameters.

3.1 Quantized Features

In order to quantize pure-power polynomial features we assume for ease of notation that M_d can be written as some power $M_d = Q^{K_d}$, where both $Q, K_d \in \mathbb{N}$. The more general case involves considering the (prime) factorization of M_d and follows the same derivation steps albeit with more intricate notation.

Definition 3.3 (Quantized Vandermonde vector)**.** For $Q, k \in \mathbb{N}$, we define the quantized Vandermonde vector $\boldsymbol{s}^{(d,k)}(\cdot) : \mathbb{C} \to \mathbb{C}^Q$ as

$$\boldsymbol{s}^{(d,k)}(x_d) := \left[1, x_d^{Q^{k-1}}, \ldots, x_d^{(Q-1)Q^{k-1}}\right].$$

The q-th element of $\boldsymbol{s}^{(d,k)}(x_d)$ is therefore

$$s^{(d,k)}(x_d)_q = (x_d)^{qQ^{k-1}}, \quad q = 0, 1, \ldots, Q - 1.$$

Theorem 3.4 (Quantized pure-power-($M_d - 1$) polynomial feature map)**.** *Each Vandermonde vector* $\boldsymbol{v}^{(d)}(x_d)$ *can be expressed as a Kronecker product of* K_d *factors*

$$\boldsymbol{v}^{(d)}(x_d) = \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}\left(x_d\right),$$

where $M_d = Q^{K_d}$*.*

Proof. From definition 3.1 we have that

$$v^{(d)}(x_d)_{m_d} = (x_d)^{m_d}.$$

Assume that $M_d = Q^{K_d}$. We proceed by tensorizing $\boldsymbol{v}^{(d)}(x_d)$ along K_d dimensions, each having size Q. Then

$$\begin{aligned} v^{(d)}(x_d)_{m_d} &= \texttt{ten}\left(v^{(d)}, Q, Q, \ldots, Q\right)_{q_1 q_2 \ldots q_{K_d}} \\ &= (x_d)^{\sum_{k=1}^{K_d} q_k Q^{k-1}} \\ &= \prod_{k=1}^{K_d} (x_d)^{q_k\, Q^{k-1}} \\ &= \prod_{k=1}^{K_d} s^{(d,k)}(x_d)_{q_k}. \end{aligned}$$

The last equality follows directly from definition 3.3. Hence by the definition of Kronecker product, we have that

$$\boldsymbol{v}^{(d)}(x_d) = \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}\left(x_d\right).$$

□

Note once more that in principle it is possible to tensorize with respect to K_d indices such that $M_d = Q_1 Q_2 \cdots Q_{K_d}$, but we restrain from doing so not to needlessly complicate notation. Theorem 3.4 allows then to quantize pure-power and Fourier features.

Corollary 3.5 (Quantized pure-power polynomials)**.** *For an input sample* $\boldsymbol{x} \in \mathbb{C}^D$*, the pure-power polynomial feature map can be expressed as*

$$\boldsymbol{z}\left(\boldsymbol{x}\right) = \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}\left(x_d\right).$$

Corollary 3.6 (Quantized Fourier feature map)**.** *For an input sample* $\boldsymbol{x} \in \mathbb{C}^D$*, the Fourier feature map can be expressed as*

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} c_d^{\frac{1}{K_d}} \boldsymbol{s}^{(d,k)}\left(e^{-\frac{2\pi j x_d}{L}}\right),$$

where $c_d = e^{2\pi\, j\, x_d\, \frac{2+M_d}{2L}}$.

Note that when quantized, both pure-power and Fourier features admit an efficient storage complexity of $\mathcal{O}(DK) = \mathcal{O}(D \log M)$ instead of $\mathcal{O}(DM)$, where $K = \max(K_1, \ldots, K_D)$.

Example 3.7. Consider $D = 2$, $M_1 = 8 = 2^3$ $M_2 = 4 = 2^2$, then the Vandermonde vector of monomials up to total degree 10 is constructed from

$$\boldsymbol{z}(\boldsymbol{x}) = [1,\, x_1] \otimes \left[1,\, x_1^2\right] \otimes \left[1,\ x_1^4\right] \otimes [1,\, x_2] \otimes \left[1,\, x_2^2\right].$$

We now present the second contribution of this article, which is the quantization of the model weights associated with quantized polynomial and Fourier features. As we will see, these quantized models are more expressive given the number of model parameters and same exact features.

4 QUANTIZED TENSOR NETWORK KERNEL MACHINES

When not considering quantization, model weights allow for tensorial indexing along the D dimensions of the inputs, i.e. $\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right)$. Corollary 3.5 and corollary 3.6 allow to exploit the Kronecker product structure of pure-power polynomial and Fourier features by further tensorizing the model weights of the tensor network-constrained kernel machines of equation (5)

$$\texttt{ten}(\boldsymbol{w},\ \underbrace{Q, Q, \ldots, Q}_{\sum_{d=1}^{D} K_d \text{ times}}).$$

These further factorized model weights can then be constrained to be a tensor network, and learned by minimizing the empirical risk in the framework of equation (5). Training a kernel machine under this constraint results in the following nonlinear optimization problem:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{n=1}^{N} \ell(\langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}\left(x_d\right), \boldsymbol{w}\rangle, y_n) + \lambda \left|\left|\boldsymbol{w}\right|\right|^2, \tag{6}$$

$$\text{s.t. } \texttt{ten}\left(\boldsymbol{w}, Q, Q, \ldots, Q\right) \text{ is a tensor network.}$$

4.1 Computational Complexity

In case of CPD, TT or TR-constrained and quantized model weights, model responses and associated gradients can be computed at the same cost as with non-quantized models:

Theorem 4.1 (Quantized tensorized kernel machine (QTKM))**.** *Consider pure-power and Fourier feature maps factorized as in corollary 3.5 and corollary 3.6 and suppose* $\texttt{ten}\left(\boldsymbol{w}, Q, Q, \ldots, Q\right)$ *is a tensor in CPD,* *TT or TR form. Then by theorem 2.3, model responses and associated gradients*

$$f_{quantized}(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}(x_d), \boldsymbol{w} \rangle,$$

can be computed in $\mathcal{O}(P)$ instead of $\mathcal{O}(\prod_{d=1}^{D} M_d)$, where $P = KDQR$ in case of CPD, and $P = KDQR^2$ in case of TT or TR.

Proof. See appendix B.2. □

Results for more general TNs can be found in appendix B.2. A graphical depiction of a QTKM can be found in figure 1b. Furthermore, when considering tensor network-specific optimization algorithms, the time complexity per iteration of training when optimizing equation (6) is lower compared to equation (5), as these methods typically optimize over a subset (typically one core) of model parameters, see appendix C.

4.2 Increased Model Expressiveness

Constraining a tensor to be a tensor network allows to distill the most salient characteristics of the data in terms of an limited number of effective parameters without destroying its multi-modal nature. This is also known as the *blessing of dimensionality* (Cichocki, 2014) and is the general underlying concept behind tensor network-based methods. In the more specific context of supervised kernel machines, these well-known empirical considerations are also captured in the rigorous framework of VC-theory (Vapnik, 1998). Khavari and Rabusseau (2021, theorem 2) have recently shown that the VC-dimension and pseudo-dimension of tensor network-constrained models of the form of equation (6) satisfies the following upper bound *irrespectively of the choice of tensor network*:

$$\mathrm{VC}(f) \leq 2P \log(12|V|),$$

where $|V|$ is the number of vertices in the TN (see definition A.1). Since quantization of the model weights increases the number of vertices in their tensor network representation, quantized models are characterized by higher upper bounds on the VC-dimension and pseudo-dimension *for the same number of model parameters*. For example, in the non-quantized case, parametrizing the TN as a CPD, TT or TR yields

$$\mathrm{VC}(f) \leq 2P \log(12D),$$

while for the quantized case

$$\mathrm{VC}(f_{\text{quantized}}) \leq 2P \log(12D \log M).$$

Hence, in case of CPD, TT and TR this additional possible model expressiveness comes at *no additional computational costs per iteration* when training with gradient descent (theorem 2.3 and 4.1). Setting $Q = 2$ provides then in this sense an optimal choice for this additional hyperparameter, as it maximizes the upper bound. In the more general case where M_d is not a power of 2, this choice corresponds with the prime factorization of M_d. It should be noted that a higher VC-dimension does not imply better performance on unseen data. However as we will see in sections 5.1 and 5.2 quantized models tend to outperform their counterparts in the underparameterized regime where TKMs are typically employed, as the gained expressiveness is put fully to good use and does not result in overfitting.

5 NUMERICAL EXPERIMENTS

In all experiments we consider a squared loss $\ell(f(\boldsymbol{x}), y) = |f(\boldsymbol{x}) - y|^2$, scale our inputs to lie in the unit box, and consider Fourier features (definition 3.2) as they notably suffer less from ill-conditioning than polynomials. In all experiments we model the weight tensor as a CPD of rank R. We do not consider other TNs in the numerical experiments for three reasons: first, it has been shown that tensor trains are more suited to model time-varying functions such as dynamical systems and time series, as opposed to CPD (Khrulkov et al., 2018). Second, CPD adds only one hyperparameter to our model as opposed to D hyperparameters for the tensor train or tensor ring. Choosing these hyperparameters (tensor train ranks) is not trivial and can yield models with very different performance for the same total number of model parameters. Third, CPD-based models are invariant to reordering of the features as opposed to tensor train. We believe that this invariance is very much desired in the context of kernel machines. We solve the ensuing optimization problem using ALS (Uschmajew, 2012). The source code and data to reproduce all experiments is available at `https://github.com/fwesel/QFF`.

5.1 Improved Generalization Capabilities

In this experiment we verify the expected quantization to positively affect the generalization capabilities of quantized models. We compare QTKM (our approach) with TKM (Wahls et al., 2014; Stoudenmire and Schwab, 2016; Kargas and Sidiropoulos, 2021; Wesel and Batselier, 2021), random Fourier features (RFF) (Rahimi and Recht, 2007), and with the full, unconstrained model (kernel ridge regression (KRR) which is our baseline, as we are dealing in all cases with squared loss). For our comparison we select eight small UCI

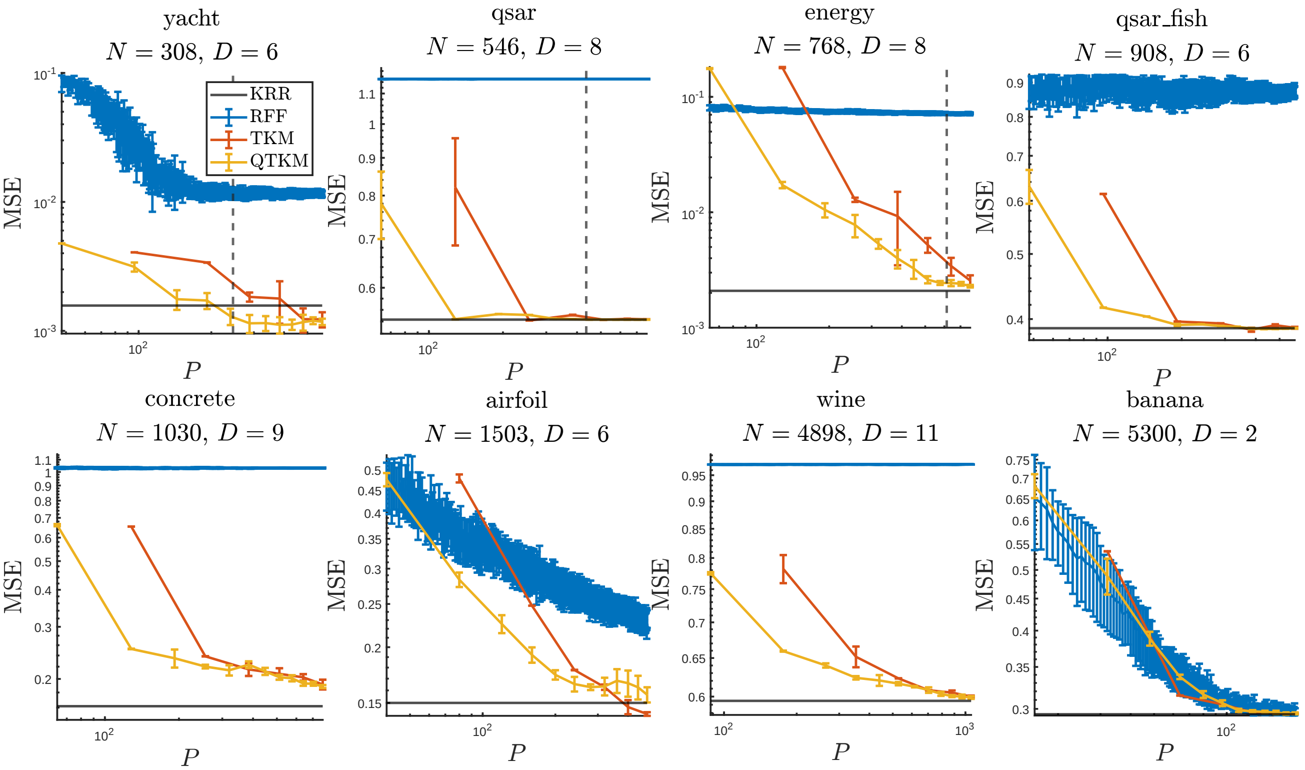


Figure 2: Plots of the test mean squared error as a function of the number of model parameters P, for different real-life datasets. In blue, random Fourier features (Rahimi and Recht, 2007), in red tensorized kernel machines with Fourier features (Wahls et al., 2014; Stoudenmire and Schwab, 2016; Kargas and Sidiropoulos, 2021; Wesel and Batselier, 2021), in yellow quantized kernel machines with Fourier features, with quantization $Q = 2$. The gray horizontal full line is the full unconstrained optimization problem, which corresponds to kernel ridge regression (KRR). The grey vertical dotted line is set at $P = N$. It can be seen that for $P < N$ case, quantization allows to achieve better generalization performance with respect to the non-quantized case.

datasets (Dua and Graff, 2017). This choice allows us to train KRR by solving its dual optimization problem and thus to implicitly consider $\prod_{d=1}^{D} M_d$ features. For each dataset, we select uniformly at random 80 % of the data for training, and keep the rest for test. We set $Q = 2$ and select the remaining hyperparameters (λ and L) by 3-fold cross validating KRR. We set the number of basis functions $M_d = 16$ uniformly for all d for all models, so that they learn from the same representation (except for RFF, which is intrinsically random). We then vary the rank R of the non-quantized tensorized model from $R = 1, 2, \ldots, 6$ and train all other models such that their number of model parameters P is at most equal to the ones of the non-quantized model. This means that for TKM $P = R\sum_{d=1}^{D} M_d$, for QTKM $P = 2R\sum_{d=1}^{D} \log_2 M_d$ and for RFF P equals the number of random frequencies. To make sure that TKM and QTKM converge, we run ALS for a very large number of iterations (5000). We repeat the procedure 10 times, and plot the mean and standard deviation of the test mean squared error (MSE) in figure 2.

In figure 2 one can observe that on all datasets, for the same number of model parameters P and identical features, the generalization performance of QTKM is equivalent or better in term of test MSE. An intuitive explanation for these results is that for equal P, quantization allows to explicitly model correlations within each of the D modes of the feature map, yielding models with increased learning capacity. We notice that while on most datasets the tensor-based approaches recover the performance of KRR, in one case, namely on the yacht dataset, the performance is better than baseline, pointing out at the regularizing effect of the quantized CPD model. Furthermore, on all datasets examined in figure 2 it can be observed that QTKM switches from underfitting to overfitting regime (first local optimum of the learning curve) before TKM, indicating that indeed its capacity is saturated with less model parameters. At that sweet spot, TKM is still underfitting and underperforming with respect to QTKM. For a further increase in model parameters both models exhibit double descent, as can be observed on the qsar, qsar_fish and airfoil datasets. Note that QTKM out-

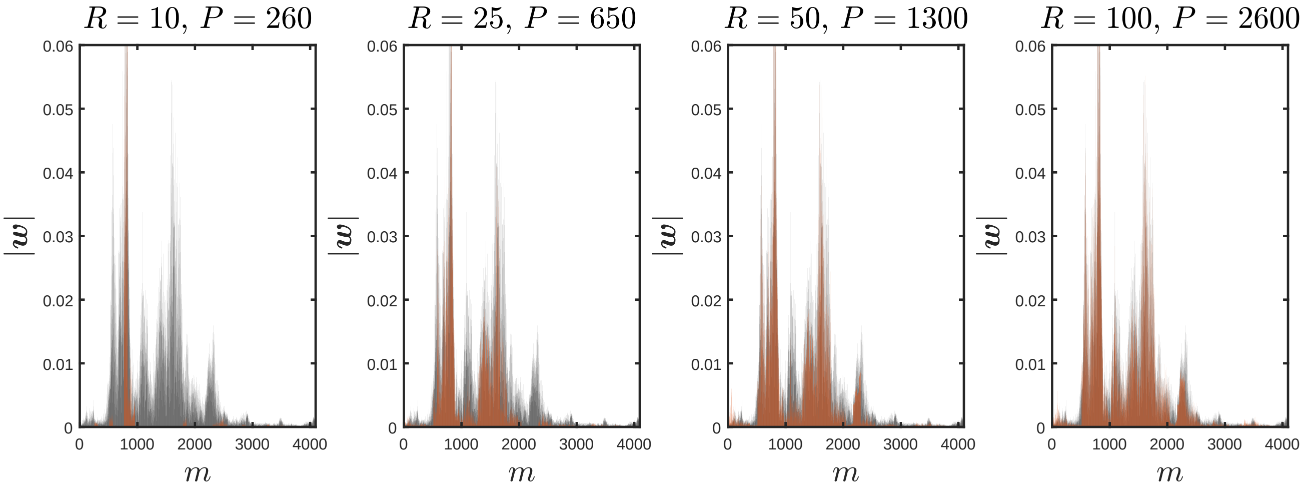


Figure 3: Sound dataset. In red, plot of the magnitude of the quantized Fourier coefficients for different values of R and total number of model parameters P. The magnitude of the full unconstrained Fourier coefficients is shown in black. It can be observed that increasing the CPD rank R recovers the peaks of frequencies with the highest magnitude.

performs TKM in a similar fashion on the training set (figure 4 in the appendix), corroborating the presented analysis. In figure 2 it can also be seen that except on the examined 2-dimensional dataset, both tensor network are consistently outperforming RFF. As we will see in section 5.2, these tensor network-based methods are able to find in a data-dependent way a parsimonious model representation given an exponentially large feature space. This is in contrast to random methods such as RFF, which perform feature selection prior to training and are in this sense oblivious to training data.

5.2 Regularizing Effect of Quantization

We would like to gain insight in the regularizing effect caused by modeling the quantized weights as an underparametrized tensor network. For this reason we investigate how the Fourier coefficients are approximated as a function of the CPD rank in a one-dimensional dataset. In order to remove other sources of regularization, we set $\lambda = 0$. The sound dataset (Wilson and Nickisch, 2015) is a one-dimensional time series regression task which comprises 60 000 sampled points of a sound wave. The training set consists of $N = 59\,309$ points, of which the remainder is kept for test. Based on the Nyquist–Shannon sampling theorem, we consider $M = 2^{13} = 8192$ Fourier features, which we quantize with $Q = 2$. We model the signal as a having unit period, hence set $L = 1$. The Fourier coefficients are modeled as a CPD tensor, with rank $R = 10, 25, 50, 100$ in order to yield underparametrized models ($P \ll M$). We plot the magnitude of the Fourier coefficients, which we obtain by minimizing equation (6) under squared loss. We compare the magnitude of the quantized weights with the magnitude of the unconstrained model response, obtained by solving equation (2), in figure 3. From figure 3 we can see that for low values of R the quantized kernel machine does not recover the coefficients associated with the lowest frequencies, as a data-independent approach would. Instead, we observe that the coefficients which are recovered for lower ranks, e.g. in case of $R = 10$, are the peaks with the highest magnitude. This is explained by the fact that the additional modes introduced by $Q = 2$-quantization force the underparametrized tensor network to model the nonlinear relation between different basis which under squared-loss maximize the energy of the signal. As the rank increases, the increased model flexibility allows to model more independent nonlinearities. We can see that already for $R = 100$ the two spectra become almost indistinguishable. We report the relative approximation error of the weights and the standardized mean absolute error on the test set in appendix D.2.

5.3 Large-Scale Regression

In order to showcase and compare out approach with existing literature in the realm of kernel machines, we consider the airline dataset (Hensman et al., 2013), an 8-dimensional dataset which consists of $N = 5\,929\,413$ recordings of commercial airplane flight delays that occurred in 2008 in the USA. As is standard on this dataset (Samo and Roberts, 2016), we consider a uniform random draw of $^2/_3 N$ for training and keep the remainder for the evaluation of the MSE on the test set and repeat the procedure ten times. In order to capture the complicated nonlinear relation between input and

Table 1: MSE for different kernel machines on the airline dataset with one standard deviation. We report the number of basis functions M per dimensions (in case of random approaches we simply report the total number of basis) and model parameters P. Notice that QTKM is able to parsimoniously predict airline delay with a restricted number of model parameters, achieving state-of-the art performance on this dataset.

Method	M	$P \downarrow$	MSE
VFF (Hensman et al., 2017)	40	320	0.827 ± 0.004
Hilbert-GP (Solin and Särkkä, 2020)	40	320	0.827 ± 0.005
VISH (Dutordoir et al., 2020)	660	660	0.834 ± 0.055
SVIGP (Hensman et al., 2013)	1000	1000	0.791 ± 0.005
Falkon (Meanti et al., 2020)	10 000	10 000	0.758 ± 0.005
TKM ($R = 4$)	64	2048	0.789 ± 0.005
TKM ($R = 6$	64	3072	0.773 ± 0.006
TKM ($R = 8$)	64	4096	0.765 ± 0.007
QTKM ($R = 20$)	64	1920	0.764 ± 0.005
QTKM ($R = 30$)	64	2880	0.754 ± 0.005
QTKM ($R = 40$)	64	3840	$\mathbf{0.748 \pm 0.005}$

output, we resort to consider $M_d = 64$ Fourier features per dimension, which we quantize with $Q = 2$. For this experiment, we set $L = 10$, $\lambda = 1 \times 10^{-10}$ and run the ALS optimizer for 25 epochs. We train three different QTKMs with $R = 20, 30, 40$.

We present the results in table 1, where we can see that QTKM (our approach) is best at predicting airline delay in term of MSE. Other grid-based approaches, such as VFE (Hensman et al., 2017) or Hilbert-GP (Solin and Särkkä, 2020), are forced to resort to additive kernel modeling and thus disregard higher-order interactions between Fourier features pertaining to different dimension. In contrast, QTKM is able to construct R data-driven explanatory variables based on an exponentially large set of Fourier features. When compared with its non-quantized counterpart TKM, we can see that our quantized approach outperforms it with approximately half of its model parameters. Training QTKM on the Intel Core i7-10610U CPU of a Dell Inc. Latitude 7410 laptop with 16 GB of RAM took (6613 ± 40) s for $R = 20$ and took $(13\,039 \pm 114)$ s for $R = 40$.

6 CONCLUSION

We proposed to quantize Fourier and pure-power polynomial features, which allowed us to quantize the model weights in the context of tensor network-constrained kernel machines. We verified experimentally the theoretically expected increase in model flexibility which allows us to construct more expressive models with the same number of model parameters which learn from the same exact features at the same computational cost per iteration.

Our approach can be readily incorporated in other tensor network-based learning methods which make use of pure-power polynomials or Fourier features.

Acknowledgments

We would like to thank the anonymous reviewers and Albert Saiapin for their numerous suggestions and improvements which have greatly improved the quality of this paper. Frederiek Wesel, and thereby this work, is supported by the Delft University of Technology AI Labs program. The authors declare no competing interests.

References

Batselier, K., Chen, Z., and Wong, N. (2017). Tensor Network alternating linear scheme for MIMO Volterra system identification. *Automatica*, 84:26–35. [2]

Blondel, M., Fujino, A., Ueda, N., and Ishihata, M. (2016). Higher-Order Factorization Machines. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc. [2]

Blondel, M., Niculae, V., Otsuka, T., and Ueda, N. (2017). Multi-output Polynomial Networks and Factorization Machines. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. [2]

Chen, Z., Batselier, K., Suykens, J. A. K., and Wong, N. (2018). Parallelized Tensor Train Learning of Polynomial Classifiers. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4621–4632. [2, 3, 15]

Cheng, S., Wang, L., and Zhang, P. (2021). Supervised learning with projected entangled pair states. *Physical Review B*, 103(12):125117. [2, 3]

Cichocki, A. (2014). Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions. *arXiv:1403.2048 [cs]*. [3, 6]

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., and Mandic, D. P. (2016). 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. [1, 2, 3, 4]

Cichocki, A., Phan, A.-H., Zhao, Q., Lee, N., Oseledets, I. V., Sugiyama, M., and Mandic, D. (2017). Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives. *Foundations and Trends® in Machine Learning*, 9(6):249–429. [3]

Comon, P., Luciani, X., and de Almeida, A. L. F. (2009). Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23(7-8):393–405. [3, 15]

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. [1]

De Branges, L. (1959). The Stone-Weierstrass Theorem. *Proceedings of the American Mathematical Society*, 10(5):822–824. [4]

De Lathauwer, L. (2008a). Decompositions of a Higher-Order Tensor in Block Terms—Part I: Lemmas for Partitioned Matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1022–1032. [3]

De Lathauwer, L. (2008b). Decompositions of a Higher-Order Tensor in Block Terms—Part II: Definitions and Uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066. [3]

Dua, D. and Graff, C. (2017). UCI Machine Learning Repository. [7]

Dutordoir, V., Durrande, N., and Hensman, J. (2020). Sparse Gaussian Processes with Spherical Harmonic Features. In *International Conference on Machine Learning*, pages 2793–2802. PMLR. [9]

Efthymiou, S., Hidary, J., and Leichenauer, S. (2019). TensorNetwork for Machine Learning. *arXiv:1906.06329 [cond-mat, physics:physics, stat]*. [2]

Evenbly, G. and Vidal, G. (2009). Algorithms for entanglement renormalization. *Physical Review B*, 79(14):144108. [3]

Favier, G. and Bouilloc, T. (2009). Parametric complexity reduction of Volterra models using tensor decompositions. In *2009 17th European Signal Processing Conference*, pages 2288–2292. [2]

Grasedyck, L. (2010). Hierarchical Singular Value Decomposition of Tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054. [3]

Hackbusch, W. and Kühn, S. (2009). A New Scheme for the Tensor Representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722. [3]

Hensman, J., Durrande, N., and Solin, A. (2017). Variational Fourier features for Gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588. [1, 4, 9]

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for Big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 282–290. AUAI Press. [8, 9]

Hitchcock, F. L. (1927). The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189. [3]

Holtz, S., Rohwedder, T., and Schneider, R. (2012). The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713. [3, 15]

Karagoz, R. and Batselier, K. (2020). Nonlinear system identification with regularized Tensor Network B-splines. *Automatica*, 122:109300. [1, 2]

Kargas, N. and Sidiropoulos, N. D. (2021). Supervised Learning and Canonical Decomposition of Multivariate Functions. *IEEE Transactions on Signal Processing*, pages 1–1. [2, 6, 7, 16]

Khavari, B. and Rabusseau, G. (2021). Lower and Upper Bounds on the Pseudo-Dimension of Tensor Network Models. In *Advances in Neural Information Processing Systems*. [3, 6, 13]

Khoromskij, B. N. (2011). O(Dlog N)-Quantics Approximation of N-d Tensors in High-Dimensional Numerical Modeling. *Constructive Approximation*, 34(2):257–280. [2]

Khrulkov, V., Novikov, A., and Oseledets, I. (2018). Expressive power of recurrent neural networks. In *International Conference on Learning Representations*. [6]

Kolda, T. G. and Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500. [2, 3, 15]

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood - Computing Hilbert Space Expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning*, pages 244–252. PMLR. [2]

Meanti, G., Carratino, L., Rosasco, L., and Rudi, A. (2020). Kernel methods through the roof: Handling billions of points efficiently. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14410–14422. Curran Associates, Inc. [9]

Meister, M., Sarlos, T., and Woodruff, D. (2019). Tight Dimensionality Reduction for Sketching Low Degree Polynomial Kernels. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. [2]

Novikov, A., Oseledets, I., and Trofimov, M. (2018). Exponential machines. *Bulletin of the Polish Academy of Sciences: Technical Sciences; 2018; 66; No 6 (Special Section on Deep Learning: Theory and Practice); 789-797.* [2, 3, 15]

Novikov, A., Rakhuba, M., and Oseledets, I. (2021). Automatic differentiation for Riemannian optimization on low-rank matrix and tensor-train manifolds. *arXiv:2103.14974 [cs, math]*. [3, 15]

Oseledets, I. V. (2011). Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317. [3]

Pham, N. and Pagh, R. (2013). Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 239–247. Association for Computing Machinery. [2]

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1177–1184. Curran Associates Inc. [2, 6, 7, 16]

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass. [4]

Rendle, S. (2010). Factorization Machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. [2]

Samo, Y.-L. K. and Roberts, S. J. (2016). String and Membrane Gaussian Processes. *Journal of Machine Learning Research*, 17(131):1–87. [8]

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press. [1]

Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E. E., and Faloutsos, C. (2017). Tensor Decomposition for Signal Processing and Machine Learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582. [2]

Solin, A. and Särkkä, S. (2020). Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30(2):419–446. [1, 4, 9]

Stoudenmire, E. M. and Schwab, D. J. (2016). Supervised learning with tensor networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4806–4814. Curran Associates Inc. [2, 3, 6, 7, 16]

Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., and Vandewalle, J. (2002). *Least Squares Support Vector Machines*. World Scientific. [1]

Tucker, L. R. (1963). Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15(122-137):3. [3]

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311. [3]

Uschmajew, A. (2012). Local Convergence of the Alternating Least Squares Algorithm for Canonical Tensor Approximation. *SIAM Journal on Matrix Analysis and Applications*, 33(2):639–652. [3, 6, 15]

Vapnik, V. N. (1998). *The Nature of Statistical Learning Theory*. Springer New York. [6]

Verstraete, F. and Cirac, J. I. (2004). Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions. [3]

Wahls, S., Koivunen, V., Poor, H. V., and Verhaegen, M. (2014). Learning multidimensional Fourier series with tensor trains. In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 394–398. [1, 2, 3, 4, 6, 7, 15, 16]

Wesel, F. and Batselier, K. (2021). Large-Scale Learning with Fourier Features and Tensor Decompositions. In *Advances in Neural Information Processing Systems*. [2, 3, 6, 7, 15, 16]

White, S. R. (1992). Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866. [3, 15]

Williams, C. and Seeger, M. (2001). Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press. [2]

Wilson, A. and Nickisch, H. (2015). Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1775–1784. PMLR. [8]

Woodruff, D. P. (2014). Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157. [2]

Zhao, Q., Zhou, G., Xie, S., Zhang, L., and Cichocki, A. (2016). Tensor Ring Decomposition. *arXiv:1606.05535 [cs]*. [3]

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes.
   (b) Complete proofs of all theoretical results. Yes.
   (c) Clear explanations of any assumptions. Yes.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes.
   (d) A description of the computing infrastructure 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, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes.
   (b) The license information of the assets, if applicable. Yes.
   (c) New assets either in the supplemental material or as a URL, if applicable. Yes. All necessary assets can be found in the anonymized URL.
   (d) Information about consent from data providers/curators. Not Applicable (UCI data).
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not applicable. All data is anonymous and open source.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. No/Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

A DEFINITIONS

Definition A.1 (Tensor network (Khavari and Rabusseau, 2021))**.** Given a graph $G = (V, E, \dim)$ where V is a set of vertices, E is a set of edges and $\dim : E \to \mathbb{N}$ assigns a dimension to each edge, a tensor network assigns a core tensor $\boldsymbol{\mathcal{C}}_v$ to each vertex of the graph, such that $\boldsymbol{\mathcal{C}}_v \in \otimes_{e \in E_v} \mathbb{C}^{\dim(e)}$. Here $E_v = \{e \in E | v \in e\}$ is the set of edges connected to vertex v. The resulting tensor is a tensor in $\otimes_{e \in E \cap V} \mathbb{C}^{\dim(e)}$. The number of parameters of the tensor network is then $P = \sum_{v \in V} \prod_{e \in E_v} \dim(e)$.

B PROOFS

B.1 Tensor Network Kernel Machine

Theorem B.1. *Suppose* $\mathtt{ten}(\boldsymbol{w}, \underbrace{M, M, \ldots, M}_{D \text{ times}})$ *is a tensor network. Then the dependency on* M *of the computational complexity for the model responses*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}(x_d), \boldsymbol{w} \rangle,$$

is $\mathcal{O}(M^t)$*, where* t *is the maximum number of singleton edges per core.*

Proof. Let t be the maximum number of singleton edges per core. Since taking the Frobenius inner product (equation (4)) involves summing over all singleton edges $\underbrace{M, M, \ldots, M}_{D \text{ times}}$, the required number of FLOPS will be $\mathcal{O}(M^t)$. □

Corollary B.2. *Suppose* $\mathtt{ten}(\boldsymbol{w}, \underbrace{M, M, \ldots, M}_{D \text{ times}})$ *is a tensor network with* $t = 1$ *maximum number of singleton edges per core. Then the dependency on* M *of the computational complexity for the model responses*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}(x_d), \boldsymbol{w} \rangle,$$

is of $\mathcal{O}(M)$*.*

Note that most used tensor networks such as CPD, Tucker, TT/TR, MERA, PEPS have $t = 1$. An example of a tensor network where t can be $t \geq 2$ or higher is hierarchical Tucker. In what follows we derive the computational complexity of the model responses of CPD and TT networks.

Theorem 2.3 (Tensorized kernel machine (TKM))**.** *Suppose* $\mathtt{ten}(\boldsymbol{w}, M_1, M_2, \ldots, M_D)$ *is a tensor in CPD, TT or TR form. Then model responses and associated gradients*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}(x_d), \boldsymbol{w} \rangle,$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = DMR$ *in case of CPD, and* $P = DMR^2$ *in case of TT or TR.*

Proof. Let $\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right)$ be a tensor in CPD form. Then

$$\begin{aligned}
f\left(\boldsymbol{x}\right) &= \langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}\left(x_d\right), \boldsymbol{w} \rangle \\
&= \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_D=0}^{M_D-1} \prod_{d=1}^{D} v_{m_d}^{(d)} \sum_{r=0}^{R-1} \prod_{d=1}^{D} w_{m_d r}^{(d)} \\
&= \sum_{r=0}^{R-1} \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_D=0}^{M_D-1} \prod_{d=1}^{D} v_{m_d}^{(d)} w_{m_d r}^{(d)} \\
&= \sum_{r=0}^{R-1} \prod_{d=1}^{D} \sum_{m_d=0}^{M_d-1} v_{m_d}^{(d)} w_{m_d r}^{(d)}.
\end{aligned}$$

Gradients can be computed efficiently by caching $\prod_{d=1}^{D} \sum_{m_d=0}^{M_d-1} v_{m_d}^{(d)} w_{r_{d-1} m_d r_d}^{(d)},\ r = 1, \ldots, R$. Hence the computational complexity of the model responses and associated gradients is of $\mathcal{O}(DMR)$. Now let $\texttt{ten}\left(\boldsymbol{w}, M_1, M_2, \ldots, M_D\right)$ be a tensor in TT/TR form. Then

$$\begin{aligned}
f\left(\boldsymbol{x}\right) &= \langle \bigotimes_{d=1}^{D} \boldsymbol{v}^{(d)}\left(x_d\right), \boldsymbol{w} \rangle \\
&= \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_D=0}^{M_D-1} \prod_{d=1}^{D} v_{m_d}^{(d)} \sum_{r_1=0}^{R_1-1} \cdots \sum_{r_D=0}^{R_D-1} \prod_{d=1}^{D} w_{r_{d-1} m_d r_d}^{(d)} \\
&= \sum_{r_1=0}^{R_1-1} \cdots \sum_{r_D=0}^{R_D-1} \sum_{m_1=0}^{M_1-1} \cdots \sum_{m_d=0}^{M_D-1} \prod_{d=1}^{D} v_{m_d}^{(d)} w_{r_{d-1} m_d r_d}^{(d)} \\
&= \sum_{r_1=0}^{R_1-1} \cdots \sum_{r_D=0}^{R_D-1} \prod_{d=1}^{D} \sum_{m_d=0}^{M_d-1} v_{m_d}^{(d)} w_{r_{d-1} m_d r_d}^{(d)},
\end{aligned}$$

which is a sequence of matrix-matrix multiplications. Gradients can be computed efficiently by caching $\sum_{m_d=0}^{M_d-1} v_{m_d}^{(d)} w_{r_{d-1} m_d r_d}^{(d)},\ r_d = 1, \ldots, R_d,\ d = 1, \ldots, D$. Hence the computational complexity of the model responses and associated gradients is of $\mathcal{O}(DMR^2)$, where $M = \max(M_1, M_2, \ldots, M_D)$ i.e. $\mathcal{O}(P)$ for both CPD and TT/TR. □

B.2 Quantized Tensor Network Kernel Machine

Theorem B.3. *Suppose* $\texttt{ten}(\boldsymbol{w}, \underbrace{Q, Q, \ldots, Q}_{DK \ \textit{times}})$ *is a tensor network. Then the dependency on* M *on the computational complexity of model responses*

$$f\left(\boldsymbol{x}\right) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K} \boldsymbol{s}^{(d,k)}\left(x_d\right), \boldsymbol{w} \rangle,$$

is of $\mathcal{O}(M^{\frac{t}{\log_Q}} \log M)$, *where* t *is the maximum number of singleton edges per core.*

Proof. Let Q be chosen such that $K = \log_Q M$. Let t be the maximum number of singleton edges per core. Taking the Frobenius inner product (equation (4)) involves summing over all singleton edges $\underbrace{Q, Q, \ldots, Q}_{D \log_Q M \ \text{times}}$. Since $Q = \frac{1}{\log_Q M}$, the required number of FLOPS will be $\mathcal{O}(Q^t \log_Q M) = \mathcal{O}(M^{\frac{s}{\log_Q}} \log M)$. □

Corollary B.4. *Suppose* $\texttt{ten}(\boldsymbol{w}, \underbrace{Q, Q, \dots, Q}_{DK\ times})$ *is a tensor network with* $t = 1$ *maximum number of singleton edges per core. Then the dependency on* M *on the computational complexity of model responses*

$$f(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}(x_d), \boldsymbol{w} \rangle,$$

is of $\mathcal{O}(\log M)$.

Theorem 4.1 (Quantized tensorized kernel machine (QTKM))**.** *Consider pure-power and Fourier feature maps factorized as in corollary 3.5 and corollary 3.6 and suppose* $\texttt{ten}(\boldsymbol{w}, Q, Q, \dots, Q)$ *is a tensor in CPD, TT or TR form. Then by theorem 2.3, model responses and associated gradients*

$$f_{quantized}(\boldsymbol{x}) = \langle \bigotimes_{d=1}^{D} \bigotimes_{k=1}^{K_d} \boldsymbol{s}^{(d,k)}(x_d), \boldsymbol{w} \rangle,$$

can be computed in $\mathcal{O}(P)$ *instead of* $\mathcal{O}(\prod_{d=1}^{D} M_d)$*, where* $P = KDQR$ *in case of CPD, and* $P = KDQR^2$ *in case of TT or TR.*

Proof. The proof follows from the proof of theorem 2.3. Since instead of summing R times over $M_1, M_2, \dots, M_D$ we are summing R times over $\underbrace{Q, Q, \dots, Q}_{DK \text{ times}}$, a model response can be evaluated in $QKDR$ FLOPS for CPD and $QKDR^2$ FLOPS for TT. Since Q is a constant which does not dependent on M and $K = \log_Q M$, we have that the computational complexities are respectively $\mathcal{O}(\log MDR)$ and $\mathcal{O}(\log MDR^2)$ for CPD and TT/TR, where $K = \log M = \max(\log M_1, \log M_2, \dots, \log M_D)$, i.e. $\mathcal{O}(P)$ for both CPD and TT/TR. □

C FASTER MULTI-CONVEX OPTIMIZATION ALGORITHMS

Quantized features allow to speedup equation (6) for a large class of multi-convex solvers such as alternating least-squares (Comon et al., 2009; Kolda and Bader, 2009; Uschmajew, 2012; Holtz et al., 2012), the density matrix renormalization group (DMRG) (White, 1992) and Riemannian optimization (Novikov et al., 2018, 2021). These solvers exploit the multi-linearity of tensor networks in order to express the empirical risk as a function of only one core of the weight tensor in tensor network form per iteration, also known as sub-problem. After solving the ensuing optimization sub-problem, this procedure is repeated for for the remaining cores, defining one epoch. The whole procedure is then repeated until convergence. When a convex quadratic loss function is used, computational benefits associated with quantization arise as it enables to solve a series of quadratic problems exactly. This is common practice in literature, see for instance Wahls et al. (2014); Chen et al. (2018); Novikov et al. (2018); Wesel and Batselier (2021).

In the exemplifying case of CPD, TT and tensor ring, for a fixed number of model parameters P, quantization allows to solve each sub-problem at a reduced computational cost of $\mathcal{O}(P^2/D^2)$ compared to a cost of $\mathcal{O}(P^2/D^2(\log M)^2)$. This yields a sub-problem complexity which is *independent* of M. A similar reduction follows for other one-layered networks. Quantifying the computational gains for other structures of tensor networks is less straightforward.

D NUMERICAL EXPERIMENTS

D.1 Improved Generalization Capabilities

We report in figure 4 the training error on the examined datasets in section 5.1. As one can observe, QTKM outperforms TKM in terms of training error (figure 4) and test error (figure 2).

D.2 Regularizing Effect of Quantization

In table 2 we repeat the number of model parameters $P = 2\log_2 MR$, the compression ratio of the quantized model weights M/P, as well as the relative approximation error of the weights $||\boldsymbol{w} - \boldsymbol{w}_{\text{CPD}}||/||\boldsymbol{w}||$ and the standardized mean absolute error (SMAE) of the reconstruction error on the test set as a function of the CPD rank.

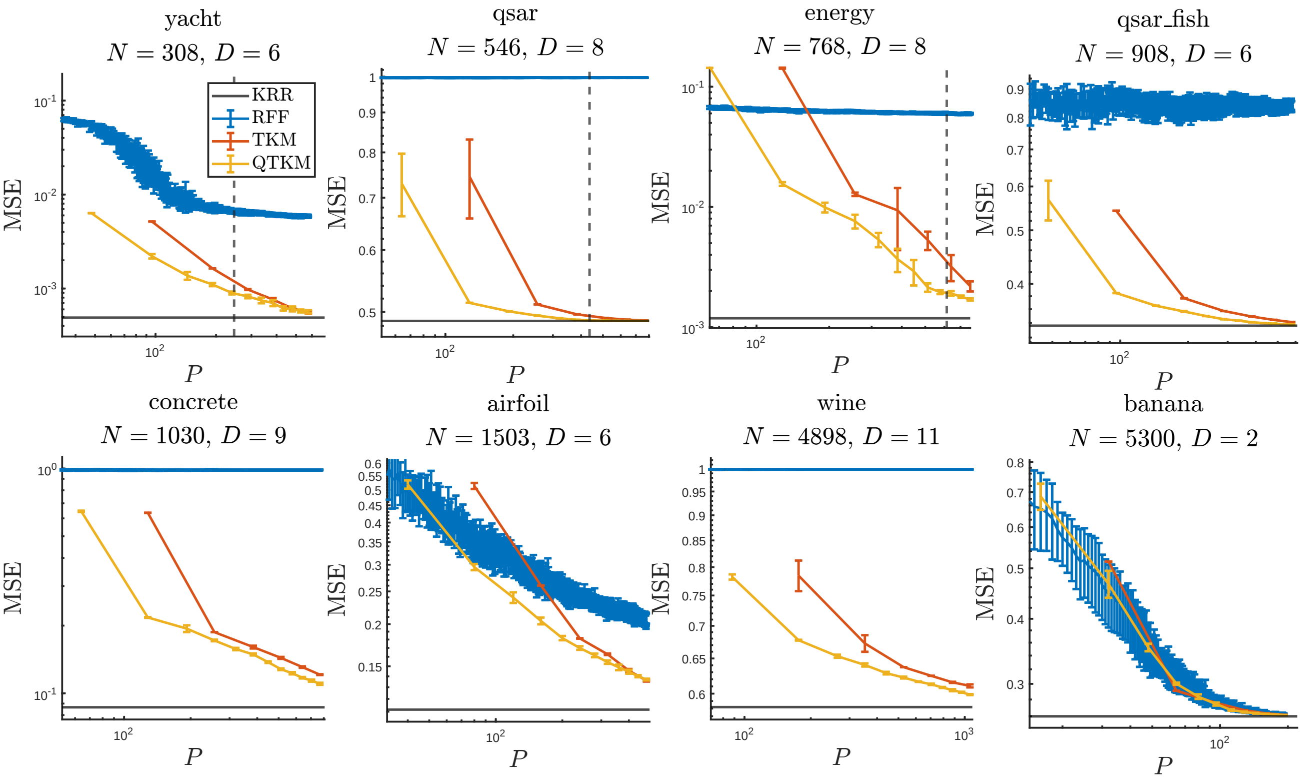


Figure 4: Plots of the train mean squared error as a function of the number of model parameters P, for different real-life datasets. In blue, random Fourier features (Rahimi and Recht, 2007), in red tensorized kernel machines with Fourier features (Wahls et al., 2014; Stoudenmire and Schwab, 2016; Kargas and Sidiropoulos, 2021; Wesel and Batselier, 2021), in yellow quantized kernel machines with Fourier features, with quantization $Q = 2$. The gray horizontal full line is the full unconstrained optimization problem, which corresponds to kernel ridge regression (KRR). The grey vertical dotted line is set at $P = N$. It can be seen that for $P < N$ case, quantization allows to achieve better performance with respect to the non-quantized case on the training set (this figure) *and* on the test set (figure 2).

Table 2: Model parameters, compression ratio and relative approximation error of the weights, and standardized mean absolute error on the test data as a function of the CPD rank.

R	P	$^{M}/_{P}$	$^{\lVert \boldsymbol{w} - \boldsymbol{w}_{\mathrm{CPD}} \rVert}/_{\lVert \boldsymbol{w} \rVert}$	SMAE
10	260	31.5	0.841	0.579
25	650	12.6	0.712	0.571
50	1300	6.3	0.528	0.451
100	2600	3.1	0.310	0.182