Cherry-Picking Gradients: Learning Low-Rank Embeddings of Visual Data via Differentiable Cross-Approximation

Mikhail Usvyatsov[1], Anastasia Makarova[1], Rafael Ballester-Ripoll[2], Maxim Rakhuba[3], Andreas Krause[1], and Konrad Schindler[1]

[1]ETH Zürich, [2]IE University, [3]HSE University

Abstract

We propose an end-to-end trainable framework that processes large-scale visual data tensors by looking at a fraction of their entries only. *Our method combines a neural network encoder with a* tensor train decomposition *to learn a low-rank latent encoding, coupled with* cross-approximation *(CA) to learn the representation through a subset of the original samples. CA is an adaptive sampling algorithm that is native to tensor decompositions and avoids working with the full high-resolution data explicitly. Instead, it actively selects local representative samples that we fetch out-of-core and on demand. The required number of samples grows only logarithmically with the size of the input. Our implicit representation of the tensor in the network enables processing large grids that could not be otherwise tractable in their uncompressed form. The proposed approach is particularly useful for large-scale multidimensional grid data (e.g., 3D tomography), and for tasks that require context over a large receptive field (e.g., predicting the medical condition of entire organs). The code is available at* https://github.com/aelphy/c-pic.

1. Introduction

Over the past decade, convolutional neural networks (CNNs) in combination with parallel processing on GPUs have brought about dramatic improvements in machine learning for image data. Unfortunately, parallel hardware is memory-limited, leading to a *curse of dimensionality*: state-of-the-art 2D network architectures are typically not viable for data with 3 or more dimensions, because one runs out of memory to store the corresponding tensors. Despite efforts to mitigate the problem via sparse convolutions [11, 18, 19] or octrees [42, 50], one must in practice limit the size of the inputs. E.g., the upper bound for 3D volumetric data is about 512^3 voxels on high-end commodity hardware.

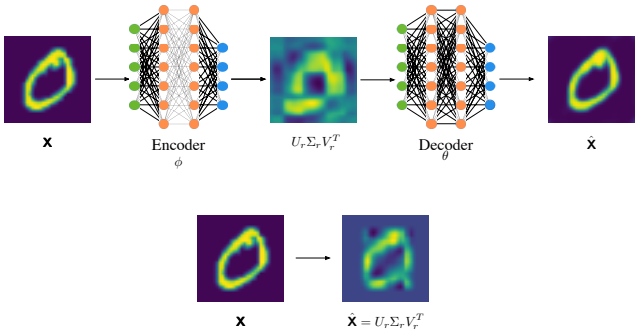


Figure 1: 2D illustration of learned low-rank embedding: rank-3 compression of the input with SVD (the matrix equivalent of TT decomposition) severely degrades the image (*bottom row*). In contrast, our encoder warps the image such that the same rank-3 truncation loses little information and can be decoded almost perfectly (*top row*).

It is well-known that visual data usually lives on lower-dimensional manifolds and, therefore, is in principle compressible; *c.f.* classical ideas like Eigenfaces [48] or non-negative tensor factorisation [44]. This motivates us to seek a more memory-efficient representation of high-dimensional visual data that is more efficient, and at the same time compatible with the gradient-based learning process of neural networks.

Selecting a sampling resolution at which data is recorded and/or processed is always a trade-off between resources (memory, run-time, power, etc.) and the level of detail and context that the algorithm has access to. Indeed, relatively small tensors are sufficient for applications where either high-frequency details are not crucial and one can operate at low spatial resolution (e.g., face recognition), or long-range context has little impact and one can process local windows (e.g., character recognition). But some tasks do require sharp details and long-range context. For instance, it has been shown that 3D object classification performance

improves with increasing resolution [19]. A similar situation arises when making holistic predictions from medical imagery: high-resolution detail helps to better spot subtle tissue changes, whereas the global context is needed to assess the extent of the condition. The ever-increasing resolution of the scanning hardware will only exacerbate this discrepancy – even current CT or MRI scanners, with typically $1024 \times 1024 \times 128$ voxels, are at the limit of what can be conveniently processed.

In this paper, we propose C-PIC (for "cherry-picking gradients"), a framework for learning with tensors while looking only at a small fraction of their entries. C-PIC exploits the fact that, after a suitable non-linear mapping, the learned representation can be constrained to have low rank. The constraint gives rise to a smart sampling strategy that adaptively selects which tensor entries to be shown to the architecture. The whole pipeline is end-to-end trainable with back-propagation, so that the learned, low-dimensional embedding is optimally tuned to a given prediction task. Crucially, our approach can operate out-of-core, meaning that it does not need to store the full input tensors in memory, but only small (hyper-)cubes around the sampled locations. It can therefore handle massive spatial resolutions that are orders of magnitude larger than the available memory, particularly on GPUs (we have experimented with volumes up to $8192^3 \approx 0.5 \cdot 10^{12}$ voxels).

The key insight underlying our novel representation is related to non-linear dimensionality reduction: if we can transform the tensor values in a way that "flattens the manifold", then we can explicitly impose a *low rank* structure on the representation, which we do with the tensor train (TT) decomposition [38]. I.e., we learn an end-to-end function that maps the input data to the desired output via a low-rank TT bottleneck. This is possible due to two important properties of the TT decomposition: *(i)* one can perform basic tensor arithmetic in the compressed format, as well as back-propagate through the decomposition; and *(ii)* there exist efficient *cross-approximation (CA)* algorithms that find an approximate TT decomposition based only a small set of samples, rather than the complete input [39]. While there have been attempts to use the TT format within a neural network [35], our work is, to the best of our knowledge, the first to employ cross-approximation for learning; making it possible to operate at high spatial resolution without running into memory limits.

With classical manifold learning, our work shares the assumption of an underlying low-dimensional, but non-linear data manifold. However, our embedding is discriminative, in the sense that the projection onto the manifold is learned end-to-end, taking into account the prediction task. In this way, the learned encoder minimises not the error when reconstructing from the latent representation, but the error of the desired output after decoding the representation. See Fig. 1 for a 2D illustration. By keeping the input and the activation maps of the encoder implicit, we circumvent what is arguably the main limitation of grid representations of dimension $D \geq 3$: their huge memory consumption, exponential in D. To summarise, our **contributions** are:

1. We design a first end-to-end neural architecture for high-dimensional, but low-rank visual data that exploits tensor decompositions;
2. We develop a computational scheme for back-propagating through cross-approximation. The differentiable CA step allows one to learns an optimal embedding from a limited number of sample evaluations and thereby opens the door to very large resolutions.
3. We develop an iterative basis projection scheme to project the learned TT features onto a canonical basis, so that they can serve as a basis for regression tasks.

We demonstrate our approach on two different medical image analysis problems and show that we perform on par with the state-of-the-art. Furthermore, C-PIC with the same hyper-parameters can work on double the resolution while other state-of-the-art methods fail due to the memory limit.

2. Background and Related Work

2.1. Tensor Train Decomposition

Tensors are a fundamental data structure for computer vision in the current age of deep learning. For our purposes, a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$ is a discrete sampling of a D-dimensional space on a grid $\mathbb{I} = I_1 \times \cdots \times I_D$, with I_d samples along dimension d.

For a long time, low-rank approximations of matrices have been used in computer vision as a tool to compress, classify, or restore visual data, e.g. [2, 45, 48, 52, 25]. However, they build on matrix-specific decomposition techniques like SVD or independent component analysis, which do not directly generalise to tensors of dimension >2.

More recently, low-rank priors have been generalized to the tensor case; see also Appendix A.1. The model used in this paper, the tensor train (TT) [38], decomposes a tensor of dimension D into D 3-dimensional tensors. Consequently, its number of parameters grows only *linearly* with the dimensionality. The TT is defined as

$$\mathbf{X}[i_1, \ldots, i_D] = \mathbf{Q}_1[1, i_1, :]\mathbf{Q}_2[:, i_2, :] \ldots \mathbf{Q}_D[:, i_D, 1], \quad (1)$$

where the tensors $\{\mathbf{Q}_d\}_{d=1}^{D}, \mathbf{Q}_d \in \mathbb{R}^{r_{d-1} \times I_d \times r_d}$, are called *TT-cores* and r_d are the *TT-ranks* ($r_0 = r_D = 1$). The TT decomposition has $\mathcal{O}\big(D \cdot (\max_d[r_d])^2 \cdot \max_d[I_d]\big)$ storage cost. Importantly, basic linear algebra operations such as linear combination of tensors can be carried out directly in this format without prior decompression (i.e., recomposing the cores).

Robust numerical schemes exist to find the TT decomposition. The standard TT-SVD algorithm yields a quasi-optimal decomposition [38] but is based on multiple rounds of singular value decomposition(SVD), i.e., it must visit all entries of the input tensor. Of crucial importance for our work is a different algorithm, known as *cross-approximation*, that efficiently constructs the TT-cores based on an adaptively chosen sequence of local samples from the input tensor. Only a small fraction of all tensor elements need to be queried; see Section 3.

2.2. Applications in Machine Learning

Tensor decompositions have been investigated as a way of extracting features from high-dimensional datasets [41, 8], and at large scale [14]. The Tucker decomposition, in particular, has recently also been extended to nonlinear interactions between the cores, with either Gaussian Processes [54] or deep neural networks [31].

[5] explore the Tucker decomposition as a lossy compression tool for multi-dimensional grid data. Our work goes further: we share the aim to compress gridded data via the low-rank representation, but learn an encoder/decoder structure tailored to the rank-constrained bottleneck to minimise the associated information loss.

In deep learning, the TT format has so far been used mostly to compress very large network layers [35]. Recently, the format was employed as part of a conditional generative model for drug design [30, 53], where a variational auto-encoder was combined with a TT-induced prior over the joint distribution of latent variables and class labels. There, a global set of TT-cores are learnable parameters, while we TT-decompose each individual input tensor, thus requiring an efficient and differentiable procedure.

2.3. Prediction of Health Indicators

In section Section 4, we demonstrate our approach on the concrete target application of predicting a patient's future condition from medical 3D scans (CT of the lung and MRI of the brain, respectively). Regressing health indicators from scan data has a long tradition in medical image analysis, e.g., [27, 51]. Following the general trend in computer vision, recent methods mostly employ deep CNNs for the task. Examples include brain age estimation from MRI scans, e.g., [24, 9]; and survival prediction from MRI scans, e.g., [26, 13]. All these works use standard CNN architectures like VGG, U-Net or ResNet, and operate on low-resolution scans (sizes below $200\times200\times100$ voxels) to stay within GPU memory limits.

3. Method

We first describe our model in feed-forward mode, where it maps tensor-valued input data to the prediction via a low-rank TT bottleneck. Then, we explain the efficient implementation and end-to-end learning of this model, including back-propagation through the cross-approximation algorithm, and a projection of TT-cores to obtain a unique feature representation.

3.1. Model Architecture

C-PIC consists of four main building blocks: *(i)* an encoder that can be seen as a learned, non-linear dimensionality reduction; *(ii)* the TT decomposition, followed by *(iii)* feature projection; and *(iv)* a conventional, learned prediction function. See Fig. 2. In the first block, a learned mapping transforms the input tensor $\mathbf{X}$ to a latent encoding $\mathbf{E}$. The low (tensor) rank of that encoding is imposed by subsequent TT decomposition. This mapping is implemented as a 3D convolutional network (but another differentiable feed-forward operator could also be used). As a result, we obtain for each location in the input tensor $\mathbf{X}$ a vector in the non-linear encoding $\mathbf{E}$, i.e., the two tensors have the same shape except for an extra channel dimension in $\mathbf{E}$.

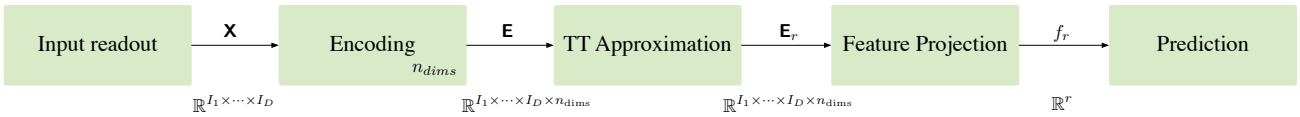
In the second block, which has not got any learnable parameters, the encoding $\mathbf{E}$ is decomposed into a set of TT-cores $\{\mathbf{Q}_d\}_{d=1}^{D+1}$ with predefined, low TT-ranks, all bounded by a hyper-parameter r. The rank r constrains the effective capacity of the representation $\mathbf{E}$ and offers a trade-off between expressiveness and memory constraints. Crucially, to build the TT decomposition one need not store the full tensors $\mathbf{X}$ and $\mathbf{E}$ in memory, rather it is sufficient to observe them at specific locations as described in Section 3.2. This makes it possible to sidestep memory limits, but poses the challenge of propagating gradients through the selection of discrete locations.

In the final two blocks, the obtained TT-cores are used as a basis for the prediction. Since the TT decomposition is not unique, they are first projected onto a canonical basis to obtain an invariant feature vector (see Section 3.3), which then serves as input for the final prediction step, in our implementation a multi-layer perceptron (MLP).

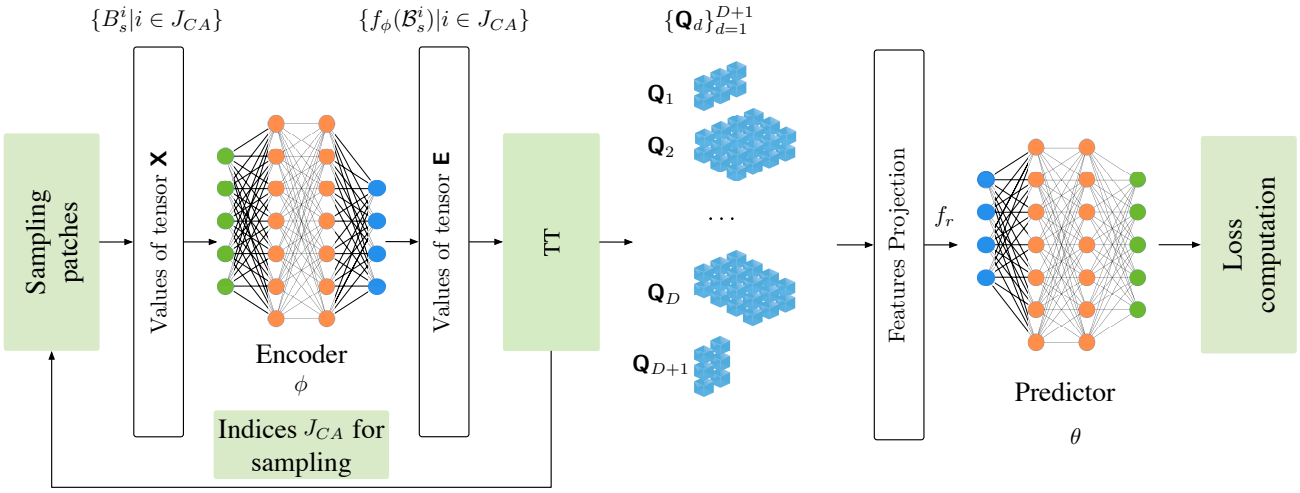
3.2. Differentiable Cross-approximation

If the tensors $\mathbf{X}$ and $\mathbf{E}$ have high resolution, storing them in memory quickly becomes intractable. Therefore, we propose to utilise an efficient *approximate* tensor learning algorithm termed cross-approximation (CA) [39]. The principle of CA is to reduce memory consumption by only considering selected entries of the tensor $\mathbf{X}$, at carefully chosen locations.

Originally, CA was conceived as a matrix sampling method [49, 7] that uses the so-called *pseudo-skeleton decomposition* [17] to approximately reconstruct a matrix $\boldsymbol{U}$ while observing only r of its rows and columns. The intersection of these rows with indices J_1 and columns with indices J_2 define an $(r\times r)$-sized submatrix $\boldsymbol{U}(J_1, J_2)$. Finding J_1, J_2 that yield the largest $|\det(\boldsymbol{U}(J_1, J_2))|$ leads to



(a) Overview of C-PIC pipeline.



(b) C-PIC with detailed view of CA.

Figure 2: General model architecture (a), and detailed view, c.f. Algorithm 1 (b). The input tensor $\mathbf{X}$ is treated as if it were partially observed. The indices J_{CA} obtained via cross-approximation define a set of locations i in $\mathbf{X}$, and the local encoder function processes a voxel-cube $\mathcal{B}_s^i$ around each of them and outputs feature vectors for the corresponding locations i in $\mathbf{E}$. These values are used to construct the TT approximation of $\mathbf{E}$.

a rank-r matrix interpolant $\boldsymbol{U}(:, J_2)\boldsymbol{U}(J_1, J_2)^{-1}\boldsymbol{U}(J_1, :)$ with the (up to a constant factor) lowest approximation error w.r.t. the original $\boldsymbol{U}$ [16].

The same idea can be applied in $D > 2$ dimensions as well: a small subset of tensor indices can, under reasonable conditions [39], be used to approximate the tensor $\mathbf{E}$, which in turn gives rise to an approximate TT decomposition $\{\mathbf{Q}_d\}_{d=1}^{D+1}$ of $\mathbf{E}$.

Let J_{CA} be a set of some N locations in the tensor $\mathbf{E}$ with $D+1$ dimensions, i.e., $J_{CA} = \{(i_1^n, \dots i_{D+1}^n)\}_{n=1}^N$. CA alternates between two steps of choosing the indices J_{CA} and building the TT-cores $\{\mathbf{Q}_d\}_{d=1}^{D+1}$ as follows:

1. *Index selection*: select a set of indices J_{CA} along all tensor dimensions, such that the approximation error is small. The error minimisation is a combinatorial problem and is in practice solved via the greedy *maxvol* heuristic [15, 43].
2. *Cross-interpolation*: compute TT-cores $\{\mathbf{Q}_d\}_{d=1}^{D+1}$ based on the entries of $\mathbf{E}$ evaluated only at indices J_{CA}. The cores $\{\mathbf{Q}_d\}_{d=1}^{D+1}$ are derived from the pseudo-skeleton reconstruction via standard matrix operations, including QR factorization, matrix multiplication, and least-squares inversion. See Appendix A.2 for further details about the CA procedure.

The value of $\mathbf{E}$ at a location $i \in J_{CA}$ is obtained by encoding the corresponding local voxel cube $\mathcal{B}_s^i$ from $\mathbf{X}$, centred at location i. In this way, one avoids having to store the full tensors in memory, instead one must only access a set $\{\mathcal{B}_s^i\}$ of N voxel cubes. The fixed, small size s of each cube determines the local context included around each sample and depends on the receptive field of the encoder, see Alg. 1.

When used to approximate a given tensor $\mathbf{E}$ in a classical way, CA iterates only over the index selection step, then explicitly assembles the TT decomposition of $\mathbf{E}$ with cross-interpolation. Our work is the first to employ CA within a larger, trainable neural architecture. This means that, during training, the source $\mathbf{E}$ changes in response to the evolving encoder weights. Consequently, the set of indices J_{CA} must also be updated throughout the learning process. While cross-interpolation consists of differentiable algebraic operations, index selection is a discrete function that poses a problem when training the pipeline end-to-end. To overcome this issue, we propose a scheme that alternates between iterative index selection and gradient descent. More specifically, we cherry-pick the tensor elements and the associated gradients as follows: First, we select and fix a set of indices J_{CA} and, using those, perform back-propagation through the cross-interpolation procedure to update the network weights. Then, to catch up with the changed encoder

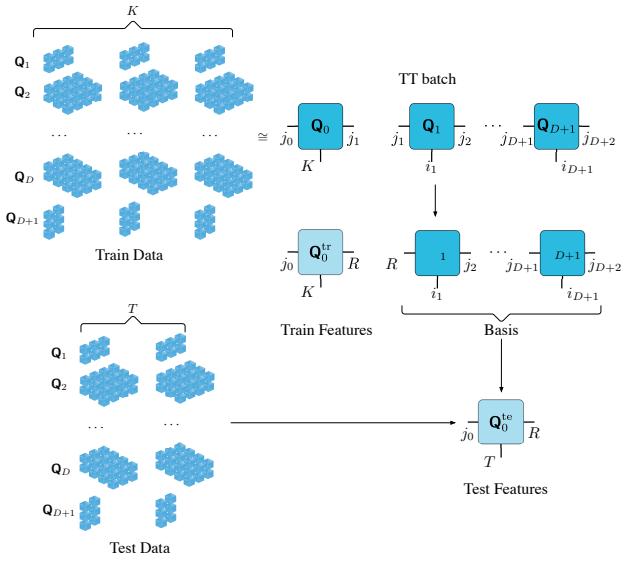


Figure 3: Feature projection. We follow the notation from [12]: each blue box represents a TT-core (3rd-order tensor). The leading and trailing dimensions satisfy $j_0 = j_{D+2} = 1$. We extract invariant features for the K training instances by stacking and rank-truncating them (like PCA for 2D matrices). This yields K feature vectors (core $\mathbf{C}_0^{tr}$) and an orthogonal basis (cores $\mathbf{C}_1, \ldots, \mathbf{C}_{D+1}$).

parameters and associated representation $\mathbf{E}$, we pick a new set of indices J_{CA}, switch back to back-propagation at those new locations, and so on. It is easy to see that this procedure converges since, for a given input, the index selection no longer changes once the encoding has converged.

Complexity of CA. It can be shown [39] that an index set J_{CA} containing $N(r) = O(Dr^2 \max_d[I_d])$ entries from $\mathbf{E}$ is sufficient to interpolate D cores, and respectively $O(Dr^2 \max_d[I_d]s)$ entries from $\mathbf{X}$. Each of the TT-cores $\{\mathbf{Q}_d\}_{d=1}^{D+1}$ contains $r^2 I_d$ elements, thus storing them does not change the memory complexity. The time complexity of the cross approximation algorithm (without the cost of sampling the tensor elements) is $O(Dr^3 \max_d[I_d])$ [39].

3.3. Feature projection

The TT decomposition is, by construction, not unique.[1] To address this issue, our pipeline includes a PCA-like step that projects the TT-cores into a canonical feature space of rank r as follows. Given multiple training instances $k = 1 \ldots K$, we view their TT decompositions $\{\{\mathbf{Q}_d^k\}_{d=1}^{D+1}\}_{k=1}^{K}$ as a set of K vectors that forms a basis. We concatenate these vectors in the TT format along a new, leading dimension to form a $(D+2)$-dimensional TT tensor $\mathbf{C}$ representing that basis, i.e. $\mathbf{C}[k, ...] \approx \mathbf{Q}_k$ (the concatenation is done in the TT compressed domain). The first core $\mathbf{C}_0$ of $\mathbf{C}$ has shape $1 \times K \times j_1$, i.e. it indexes the training instances along its spatial dimension. Next, we orthogonalise $\mathbf{C}$ with respect to $\mathbf{C}_0$ and rank-truncate the resulting core into a $K \times r$ feature matrix $\mathbf{C}_0^{\text{tr}}$. The trailing cores $\{\mathbf{C}_d\}_{d=1}^{D+1}$ now form an orthogonal basis, while matrix $\mathbf{C}_0^{\text{tr}}$ contains one r-dimensional feature vector $\boldsymbol{f}_k$ for each input $\mathbf{X}_k$ that is now invariant to the choice of coefficients in the TT representation $\mathbf{Q}_k$. The whole procedure is an extension of standard PCA matrix projection to the case where basis elements are TT tensors; see also Fig. 3. For inference, we similarly concatenate input instances into a new tensor, which we then project onto the learned basis to obtain their corresponding r-sized feature vectors. We refer to the supplementary material for further details.

[1] E.g., one can create an equivalent TT with different weights by right-multiplying all slices of some core with any non-singular matrix $\boldsymbol{R}$ and left-multiplying all slices of the subsequent core with its inverse $\boldsymbol{R}^{-1}$.

Algorithm 1 DIFFERENTIABLE CA FOR TT

$\mathbb{I} = I_1 \times I_2 \times I_3$ – 3D grid
$\mathbf{X} \in \mathbb{R}^{\mathbb{I}}$ – input visual data
$\mathbf{E} \in \mathbb{R}^{\mathbb{I} \times n_{\text{dims}}}$ – full-rank output of encoder f_ϕ
$\mathcal{B}_s^i \subset \mathbb{I} \times n_{\text{dims}}$ – s-neighborhood of $i \in \mathbb{I} \times n_{\text{dims}}$

Require: Input data $\mathbf{X}$, local size s
for epoch $= 1, \ldots, n_{\text{epochs}}$ **do**
  for $d = 1, 2, 3, 4$ **do**
    Select CA indices $J_{CA} \subset \mathbb{I} \times n_{\text{dims}}$ (Sec. 3.2)
  end for
  for $d = 1, 2, 3, 4$ **do**
    Get $\mathbf{X}[j], \ \forall j \in \{\mathcal{B}_s^i | \ \forall i \in J_{CA}\}$
    Get $\mathbf{E}[i] = f_\phi(\mathcal{B}_s^i), \ \ \forall i \in J_{CA}$
    Compute $\mathbf{Q}_d$ via *cross-interpolation* (Sec. 3.2)
  end for
  Project cores $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \mathbf{Q}_4$ into lower-dimensional features f_r (Sec. 3.3)
  Compute loss l of f_r
  Update cores via backprop(l)
end for
Output: $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3, \mathbf{Q}_4$

3.4. Technical Details

Tensorisation. An important technical detail along the way is the shape of the embedding $\mathbf{E}$ that affects the memory complexity. In principle, one can directly apply TT decomposition to the tensor $\mathbf{E}$ sampling and storing $O(Dr^2 \max_d[I_d]s)$ entries of it. However, if the tensor has high spatial resolution, i.e., $\max_d[I_d]$ is large, one can reach better memory complexity by employing the so-called Quantised[2] Tensor Train (QTT) decomposition [28, 37].

The idea of QTT is to build a TT decomposition for

[2] The name does not imply quantisation of real-valued tensor entries.

the tensor after reshaping it to a higher dimensional one. Particularly, if all $\{\log_2 I_d\}_{d=1}^{D}$ are natural numbers, a D-dimensional tensor $\mathbf{E}$ with sizes $\{I_d\}_{d=1}^{D}$ can be reshaped into a D'-dimensional tensor $\tilde{\mathbf{E}}$ with $D' = \sum_d \log_2 I_d$ and sizes $\{I_d = 2\}_{d=1}^{D'}$. As the result, QTT decomposition of $\mathbf{E}$ requires a storage cost of $\mathcal{O}(r^2 D \max_d[\log_2 I_d]s)$, as opposed to the initial $\mathcal{O}(r^2 D \max_d[I_d]s)$. Intuitively, the QTT scheme exploits the similarity between adjacent voxels in the uncompressed tensor $\mathbf{E}$ and is related to the wavelet transform; see, e.g., [36]. Note that the reshaping is only done locally and implicitly within the QTT routine, by a function that maps index tuples from $\mathbf{E}$ to $\tilde{\mathbf{E}}$ and vice versa.

QTT is the most sample-efficient scheme for tensors with large $\max_d[I_d]$ and we exploit it in Section 4 to handle resolutions that are intractable with standard deep learning models. Still, our scheme is flexible. If the number of samples is not a concern, one can use the conventional TT representation without reshaping during CA. In principle, it is also possible to use our scheme with the exact TT-SVD algorithm instead of the approximate CA to find the decomposition, if the inputs are small enough to fit them into memory.

Feature projection and batching. For PCA, the number of samples K must be at least as large as the feature dimension r. Consequently, the batch size during learning must be at least r samples per mini-batch. Note that the common basis would in principle have to be computed over all training samples. In practice we cache the basis in each mini-batch and append it to the cores of the next mini-batch to converge towards a stable, common basis at the end of an epoch.

Numerical issues. The basis computation is implemented with the *cvxpylayer* method [1], which we found to have better stability than other algebraic schemes during the backward pass through our differentiable CA. Due to the memory-efficiency of C-PIC we are also able to train it with *float64* precision to further improve numerical stability. We do this in all experiments (baselines had to be trained with *float32* to stay within memory limits).

Index batching. A subtle technical detail is that performing the backward pass simultaneously for all indices selected by CA still requires significant memory, especially for large inputs that require more CA samples. To further reduce memory consumption, one can switch to batch-wise processing of the CA indices, such that only the gradients for one batch must be held in memory. However, the price to pay is an increase in runtime, proportional to the number of batches, as one has to run the cross-interpolation step more often. We have implemented the index batching trick and have empirically verified convergence for tensors up to size 8192^3. Still, we recommend to use index batching only when necessary, as it greatly slows down the training (and even for scans of size 512^3 the complete backward pass fits into the memory of a modern GPU).

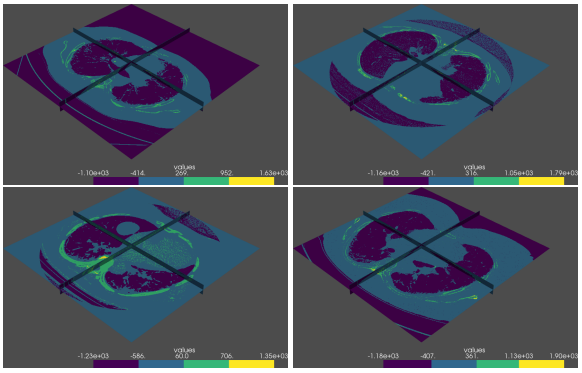


Figure 4: Examples from OSIC, resolution $32 \times 512 \times 512$.

4. Experiments

To illustrate the effectiveness of C-PIC, we apply it to two different prediction tasks where health indicators shall be regressed from medical 3D scans. The tasks were selected because of their global, holistic nature, i.e., in both cases one should assess the state of an entire organ and the future progression of the condition, for which it makes sense to process the entire scan, rather than cut it into smaller tiles.

4.1. Datasets

OSIC Pulmonary Fibrosis Progression is a dataset of CT scans of patients' lungs, originally released for a Kaggle competition [40]. Example scans are shown in Fig. 4. Pulmonary fibrosis causes a progressive decline of the pulmonary capacity, and the goal of the challenge is to predict that decline from a scan taken at time $t = 0$. Lung capacity is quantified by forced vital capacity (FVC, the volume of exhaled air exhaled). For the patients in the dataset it has been repeatedly measured over 1-2 years after the scan by means of a spirometer. FVC as a function of time (in weeks) is the regression target. Overall, there are 176 patients and 1549 individual ground truth FVC values. As error metric, the creators of the challenge proposed the modified Laplace Log Likelihood (mLLL), defined as $\text{mLLL} = -\sqrt{2}\Delta/\sigma - \ln(\sqrt{2}\sigma)$; with σ the standard deviation of the predicted FVC, truncated at 70 FVC units, and Δ the absolute error of the predicted FVC, truncated at 1000 FVC units. For training we use quantile regression. From predicted $\{0.2, 0.5, 0.8\}$ quantiles we compute both predicted FVC and standard deviation.

MICCAI 2020 BraTS is a dataset of MRI scans [34] showing brains with a specific type of tumor [33, 3, 4]. Examples are shown in Fig. 5. The target value that should be predicted from a scan is the patient's survival time (in days) after the scan was taken. The participants of the study are divided into two groups, where the first group underwent a specific type of treatment (gross total resection surgery),

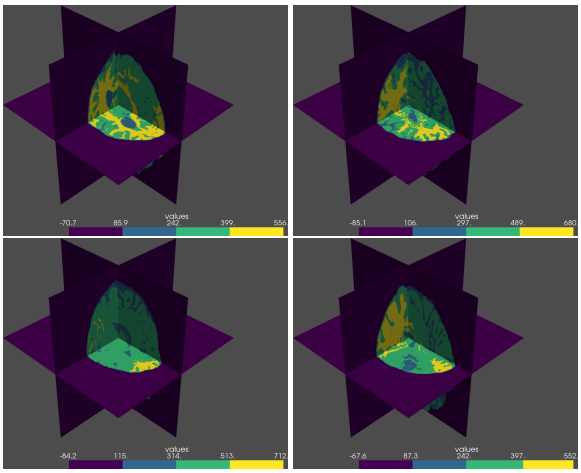


Figure 5: Examples from BraTS, resolution $256 \times 256 \times 256$.

whereas the second group was not. In total there are for 235 patients. We discard grouping and treat all scans as one single dataset for survival prediction. As error metric, we use the RMSE of the predicted survival time. During training we normalise the survival time to 5 years.

Synthetic upsampling. The goal of our work is efficient, compressed representation learning that is able to handle large, high-resolution data. However, there do not seem to be any high-resolution benchmark datasets of sufficient size (although modern scanners can capture up to at least $1024 \times 1024 \times 128$ voxels). Therefore, we also synthetically upsample the two datasets to $2\times$ higher resolution along each dimension with 3^{rd}-order spline interpolation, to obtain $8\times$ higher voxel count. Clearly, this step does not add any information to the lower-resolution originals, so we do not expect better performance, still the upsampled version gives us an opportunity to verify that our approach can indeed handle such large volumes. In fact, it does so without any loss of accuracy, which supports the hypothesis that the data has low rank and can therefore be compressed without information loss.

Implementation Details. The detailed layer structure of the CNN encoder and the MLP for prediction are given in Appendix A.4. All models are trained with RAdam [32], with base learning rate 10^{-3}. All quantitative results are averages over five-fold cross-validation.

4.2. Results

We first apply C-PIC to the data in its original form (without upsampling), and compare it to a 3D version of ResNet-34 [21] as a baseline. C-PIC is trained with batch size 20, for the baseline we had to reduce the batch size to 2 to fit the training into memory. To show that the CNN encoder is indeed needed before the low-rank constraint can be imposed, we also run our pipeline without the encoder. I.e., the raw input tensor $\mathbf{X}$ is fed into TT decomposition, projected to a canonical feature vector, and fed into the regression MLP.

Quantitative results for the **OSIC** dataset are shown in Table 1. They show that C-PIC, with rank $r = 10$ and channel depth $n_{\text{dim}} = 16$, not only needs a lot less memory, but in fact predicts FVC significantly better than the ResNet baseline. The performance gain provides evidence that the low-rank assumption underlying our method is justified, at least for the medical scan data we have used: if the intrinsic rank of the data were higher, there would have to be at least some performance loss; whereas if the assumption is valid, it can even act as a regulariser for the learning process. The TT+MLP baseline, on the contrary, performs a bit worse than ResNet and significantly worse than C-PIC, i.e., there appears to be a clear benefit in non-linearly transforming the scans to a "TT-friendly" representation with the convolutional encoder, and consequently in the associated end-to-end learning framework.

As a next step, we perform the same experiment with the up-scaled scans to see how our method scales to larger volumes. At the increased size of $64 \times 1024 \times 1024$ voxels the ResNet baseline can no longer be trained, as even a last-generation GPU with 24GB of on-board RAM runs out of memory already with batch size 1. On the contrary, C-PIC reaches the same performance as for the smaller scans (as mentioned earlier, no improvement is expected, since the synthetic up-sampling, in contrast to actual high-resolution scanning, does not add information). The table also shows that the huge memory savings of C-PIC of course come at the price of longer training time because of the added complexity to back-propagate through the TT bottleneck and CA algorithm. The difference is partly due to our implementation being not nearly as optimised as standard back-propagation code; but we cannot at this point quantify the speed-up achievable with a careful implementation. Note, however, the training time grows sub-linearly with the resolution, due to the favourable scaling properties of CA.

Results for **BraTS** are shown in Table 2. For the bigger scan volume and more complex image content of the brain scans, we keep the rank $r = 10$, but increase the channel depth of the encoding to $n_{\text{dim}} = 32$ as a default. Again, C-PIC matches the performance of ResNet baseline, with greatly reduced memory consumption. In fact, it even reaches a slightly lower RMSE, but in this case the margin is small and we do not claim to outperform the baseline. Additionally, the table also shows the impact of different channel depths in the encoder. Too few channels negatively affect the prediction, whereas too many significantly increase the runtime. We emphasise that, while adding channels in the latent space increases the representation power of the encoding $\mathbf{E}$, it only adds a tiny number of weights (for the corresponding convolution kernels). The added channels can

Table 1: OSIC Pulmonary Fibrosis Progression results. C-PIC outperforms the baselines, and can also handle 8× larger scan volumes, contrary to a 3D ResNet (marked as N/A in the table).

	resolution	mLLL↑	training time	prediction time	fw/bw memory	# params
ResNet 34	$32 \times 512 \times 512$	-6.86	4650 s. / epoch	0.2 s. / sample	7.0 Gb	67M
ResNet 34	$64 \times 1024 \times 1024$	N/A	N/A	N/A	57.9 Gb	67M
TT + MLP	$32 \times 512 \times 512$	-6.91	27534 s. / epoch	14.9 s. / sample	1.0 Gb	64K
C-PIC	$32 \times 512 \times 512$	**-6.73**	51480 s. / epoch	25.2 s. / sample	3.5 Gb	87K
C-PIC	$64 \times 1024 \times 1024$	**-6.73**	62478 s. / epoch	46.1 s. / sample	4.2 Gb	87K

Table 2: MICCAI 2020 BraTS results. C-PIC outperforms the baseline in terms of RMSE of the predicted survival time. Additionally, the table also shows C-PIC results with different channel depth of the encoding $\mathbf{E}$. Reducing the channel depth too far hurts performance, even with the same tensor rank $r = 10$.

	resolution	RMSE↓	training time	prediction time	fw/bw memory	# params
ResNet 34	256^3	48.7 days	519 s. / epoch	0.3 s. / sample	14.0 Gb	67M
TT + MLP	256^3	83.9 days	646 s. / epoch	2.9 s. / sample	4.2 Gb	3K
C-PIC $n_{dim} = 32, r = 10$	256^3	**48.2** days	3300 s. / epoch	13.4 s. / sample	8.9 Gb	37K
C-PIC $n_{dim} = 16, r = 10$	256^3	49.1 days	2979 s. / epoch	12.8 s. / sample	8.7 Gb	27K
C-PIC $n_{dim} = 8, r = 10$	256^3	51.1 days	2883 s. / epoch	12.1 s. / sample	7.9 Gb	21K
C-PIC $n_{dim} = 8, r = 10$	512^3	51.2 days	16560 s. / epoch	79.0 s. / sample	45.5 Gb	21K
C-PIC $n_{dim} = 8, r = 12$	256^3	51.1 days	5520 s. / epoch	27.9 s. / sample	13.4 Gb	21K
C-PIC $n_{dim} = 8, r = 15$	256^3	51.1 days	7140 s. / epoch	35.8 s. / sample	18.6 Gb	22K

be interpreted as additional dimensions of the encoded data manifold, which make it easier to "flatten". They do not relax the low-rank constraint: independent of the number of channels in its last dimension, the tensor $\mathbf{E}$ is decomposed into cores $\{\mathbf{Q}_d\}$ with the same tensor rank $r = 10$.

We also test the influence of the tensor rank r on performance, with fixed, low $n_{\text{dim}} = 8$. For ranks $r \in \{10, 12, 15\}$ we observe similar performance. With values $r < 10$ the training tends to become unstable, thus preventing the model from learning. Whereas for $r > 20$ the training went out of memory (but our implementation is not fully optimised, so higher ranks are likely possible).

5. Conclusion

We have developed a neural network architecture that includes the truncated tensor-train decomposition as a low-rank latent representation, and have devised methods to back-propagate through the decomposition. Most notably, we have shown how the compressed TT encoding can be learned by cross-approximation, from a sparse set of local samples drawn from suitable locations of the input tensor. Thanks to this strategy, there is no need to store the uncompressed input tensor explicitly, which in turn makes it possible to process large, high-dimensional grids that exceed the memory of commodity hardware. In experiments on medical CT and MRI scans, we have demonstrated that our C-PIC method matches or even exceeds the performance of a conventional CNN regressor; while using orders of magnitude less memory, thus making it possible to process much larger data volumes, which we expect to increasingly see in the near future as scanning hardware improves. While we have, for practical reasons, concentrated on 3D scan data, our method is generic. As long as the requirement of low tensor rank is met (after a non-linear encoding tuned to fit the subsequent decomposition), our method can also be utilised with tensorial data of dimension >3.

A limitation of C-PIC is that TT decomposition is not robust against translations and rotations of the input data space, i.e., the inputs are implicitly assumed to be roughly aligned (like medical scans). We do not expect it to work as well for arbitrarily shifted and/or rotated inputs, unless the encoder can compensate for such transformations. One possible solution is to actively favour invariance of the encoding during training, for instance by deep supervision or suitable data augmentation. We leave this for future work.

In this work we have experimented only with regression tasks. However, the low-rank latent embedding that we learn should be equally applicable in combination with other tasks, like classification or segmentation. We speculate that it may even serve as a basis for a generative model.

References

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *NeurIPS*, 2019.

[2] H. Andrews and C. Patterson. Singular value decompositions and digital image processing. *IEEE TASSP*, 24(1):26–53, 1976.

[3] Spyridon Bakas, Hamed Akbari, Aristeidis Sotiras, Michel Bilello, Martin Rozycki, Justin S Kirby, John B Freymann, Keyvan Farahani, and Christos Davatzikos. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. *Scientific Data*, 4(1):1–13, 2017.

[4] Spyridon Bakas, Mauricio Reyes, Andras Jakab, Stefan Bauer, Markus Rempfler, Alessandro Crimi, Russell Takeshi Shinohara, Christoph Berger, Sung Min Ha, Martin Rozycki, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge. *arXiv preprint arXiv:1811.02629*, 2018.

[5] Rafael Ballester-Ripoll, Peter Lindstrom, and Renato Pajarola. TTHRESH: Tensor compression for multidimensional visual data. *IEEE TVCG*, 26(9):2891–2903, 2019.

[6] Rafael Ballester-Ripoll, Susanne K. Suter, and Renato Pajarola. Analysis of tensor approximation for compression-domain volume visualization. *Computers & Graphics*, 47:34–47, 2015.

[7] Mario Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[8] Johann A Bengua, Ho N Phien, and Hoang D Tuan. Optimal feature extraction and classification of tensors via matrix product state decomposition. In *2015 IEEE Big Data*, 2015.

[9] Camilo Bermudez, Andrew J.Plassard, Shikha Chaganti, Yuankai Huo, Katherine S. Aboud, Laurie E. Cutting, Susan M. Resnick, and Bennett A. Landman. Anatomical context improves deep learning on the brain age estimation task. *Magnetic Resonance Imaging*, 62:70–77, 2019.

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

[11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[12] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

[13] Xue Feng, Nicholas J Tustison, Sohil H Patel, and Craig H Meyer. Brain tumor segmentation using an ensemble of 3d U-nets and overall survival prediction using radiomic features. *Frontiers in Computational Neuroscience*, 14:25, 2020.

[14] Krzysztof Fonał and Rafał Zdunek. Distributed and randomized tensor train decomposition for feature extraction. In *IJCNN*, 2019.

[15] Sergei A Goreinov, Ivan V Oseledets, Dimitry V Savostyanov, Eugene E Tyrtyshnikov, and Nikolay L Zamarashkin. How to find a good submatrix. In Vadim Olshevsky and Eugene E Tyrtyshnikov, editors, *Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub*, pages 247–256. World Scientific, 2010.

[16] S. A. Goreinov and Eugene E. Tyrtyshnikov. The maximal-volume concept in approximation by low-rank matrices. *Contemporary Mathematics*, 280:47–51, 2001.

[17] Sergei A Goreinov, Evgeny E Tyrtyshnikov, and Nickolai L Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261(1-3):1–21, 1997.

[18] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

[19] Timo Hackel, Mikhail Usvyatsov, Silvano Galliani, Jan D Wegner, and Konrad Schindler. Inference, Learning and Attention Mechanisms that Exploit and Preserve Sparsity in CNNs. *IJCV*, 2020.

[20] Richard A Harshman. Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 1970.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[22] F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *J Mathematical Physics*, 7(1):39–79, 1927.

[23] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):683–713, Mar. 2012.

[24] T. Huang, H. Chen, R. Fujimoto, K. Ito, K. Wu, K. Sato, Y. Taki, H. Fukuda, and T. Aoki. Age estimation from brain MRI images using deep learning. In *ISBI 2017*, 2017.

[25] Piotr Indyk. Learning-based low-rank approximations. In *NeurIPS*, 2019.

[26] Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H Maier-Hein. Brain tumor segmentation and radiomics survival prediction: Contribution to the BraTS 2017 challenge. In *MICCAI Brain Lesion Workshop*, 2017.

[27] Anna K Jerebko, James D Malley, Marek Franaszek, and Ronald M Summers. Multiple neural network classification scheme for detection of colonic polyps in CT colonography data sets. *Academic Radiology*, 10(2):154–160, 2003.

[28] Boris N Khoromskij. O (dlog n)-quantics approximation of n-d tensors in high-dimensional numerical modeling. *Constructive Approximation*, 34(2):257–280, 2011.

[29] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[30] Maksim Kuznetsov, Daniil Polykovskiy, Dmitry Vetrov, and Alexander Zhebrak. Subset-conditioned generation using variational autoencoder with a learnable tensor-train induced prior. *NeurIPS Workshops*, 2018.

[31] Bin Liu, Lirong He, Shandian Zhe, Yingmin Li, and Zenglin Xu. DeepCP: Flexible nonlinear tensor decomposition. In *NeurIPS Workshops*, 2017.

[32] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, April 2020.

[33] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE TMI*, 34(10):1993–2024, 2014.

[34] MICCAI 2020 Brain Tumor Segmentation (BraTS) Challenge. http://braintumorsegmentation.org, 2020.

[35] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *NeurIPS*, 2015.

[36] Ivan Oseledets and Evgeny E. Tyrtyshnikov. Algebraic wavelet transform via quantics tensor train decomposition. *SIAM J on Scientific Computing*, 33(3):1315–1328, 2011.

[37] Ivan V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM J Matrix Anal Appl*, 31(4):2130–2145, 2010.

[38] Ivan V. Oseledets. Tensor-train decomposition. *SIAM J on Scientific Computing*, 33(5):2295–2317, 2011.

[39] Ivan V. Oseledets and Evgeny E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[40] OSIC Pulmonary Fibrosis Progression – predict lung function decline. https://www.kaggle.com/c/osic-pulmonary-fibrosis-progression, 2020.

[41] Anh Huy Phan and Andrzej Cichocki. Tensor decompositions for feature extraction and classification of high dimensional datasets. *Nonlinear Theory and its Applications*, 1(1):37–68, 2010.

[42] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3D representations at high resolutions. In *CVPR*, 2017.

[43] Dmitry V Savostyanov. Quasioptimality of maximum-volume cross interpolation of tensors. *Linear Algebra and its Applications*, 458:217–244, 2014.

[44] Amnon Shashua and Tamir Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *ICML*, 2005.

[45] L Sirovich and M Kirby. Low-dimensional procedure for the characterization of human faces. *JOSA A*, 4:519–524, 1987.

[46] Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15:122–137, 1963.

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

[48] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[49] Eugene Tyrtyshnikov. Incomplete cross approximation in the mosaic-skeleton method. *Computing*, 64(4):367–380, 2000.

[50] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3d shape analysis. *ACM TOG*, 36(4):1–11, 2017.

[51] Lifeng Yang, Jingbo Yang, Xiaobo Zhou, Liyu Huang, Weiling Zhao, Tao Wang, Jian Zhuang, and Jie Tian. Development of a radiomics nomogram based on the 2D and 3D CT features to predict the survival of non-small cell lung cancer patients. *European Radiology*, 29(5):2196–2206, 2019.

[52] Yangmuzi Zhang, Zhuolin Jiang, and Larry S. Davis. Learning structured low-rank representations for image classification. In *CVPR*, 2013.

[53] Alex Zhavoronkov, Yan A Ivanenkov, Alex Aliper, Mark S Veselov, Vladimir A Aladinskiy, Anastasiya V Aladinskaya, Victor A Terentiev, Daniil A Polykovskiy, Maksim D Kuznetsov, Arip Asadulaev, et al. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nature Biotechnology*, 37(9):1038–1040, 2019.

[54] Shandian Zhe, Kai Zhang, Pengyuan Wang, Kuang-Chih Lee, Zenglin Xu, Yuan Qi, and Zoubin Ghahramani. Distributed flexible nonlinear tensor factorization. In *NeurIPS*, 2016.