# Matrix and tensor methods for efficient compression and inference in deep neural networks

Ivan Oseledets

CEO, AIRI; Professor and Lab Head @Skoltech

# AIRI: leading AI Institute in Russia

- 240 employees
- Top-1 on A*/A publications in Russia
- 21 teams on all direction on AI (both fundamental and applied)
- **Big projects**: AGI for Medicine; AI for Drug Design (AIDD); FusionBrain (Multimodal AI); Quantum
- Research directions: New Materials, Efficient Algorithms, Generative AI, Self-Supervised learning, …. https://airi.net/

# Tensor decompositions: basics (1)

- We have a d-dimensional array $A(i_1, \ldots, i_d)$
- The representation suffers from the curse of dimensionality
- In many applications, we can replace/approximate a tensor using the idea of separation of variables
- Main formats: canonical format, Tucker format, tensor train decomposition, H-Tucker format

$$A(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \ldots U_d(i_d, \alpha)$$

$$A(i_1, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_d} G(\alpha_1, \ldots, \alpha_d) U_1(i_1, \alpha_1) \ldots U_d(i_d, \alpha_d)$$

$$A(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d)$$

# Tensor decompositions: basics (2)

- Canonical format is not always easy to compute
- Tucker format works for small dimensions
- TT/HT formats can be computed using stable algorithms, vast literature exists on this

$$A(i_1, \ldots, i_d) = \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha) \ldots U_d(i_d, \alpha)$$

$$A(i_1, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_d} G(\alpha_1, \ldots, \alpha_d) U_1(i_1, \alpha_1) \ldots U_d(i_d, \alpha_d)$$

$$A(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d)$$

**Tensor-train decomposition**

IV Oseledets - SIAM Journal on Scientific Computing, 2011 - SIAM

… the infimum is taken over all **tensor trains** with TT-ranks bounded by rk. Then, by the definition of the infimum, there exists a sequence of **tensor trains** B … All elements of the **tensors** B …

Save Cite Cited by 2373 Related articles All 9 versions

# Important properties of Tensor train-decomposition

- Quasi-optimal approximation can be computed via sequence of SVD
- We can recover a low-rank tensor from elements exactly (so-called cross approximation)
- Efficient optimization is possible using Riemannian optimization

$$A(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d)$$

# Compression of convolutional networks using tensors

- The paper by Lebedev, Rakhuba, Ganin, Lempitsky and O. was the first paper which proposed to use CP-decomposition to represent filters in CNN
- Such decompositions later motivated new architectures with 1x1 and depth-wise separable convolutions
- Approximate, the fine-
- Several successful Huawei projects for CNN compression

$$V(x, y, t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K(i - x + \delta, j - y + \delta, s, t) U(i, j, s)$$

$$K(i, j, s, t) = \sum_{r=1}^{R} K^x(i - x + \delta, r) K^y(j - y + \delta, r) K^s(s, r) K^t(t, r)$$

Speeding-up **convolutional neural networks** using fine-tuned cp-decomposition
V Lebedev, Y Ganin, M Rakhuba, I **Oseledets**… - arXiv preprint arXiv …, 2014 - arxiv.org
… speeding up **convolution** layers within large **convolutional neural networks** based on **tensor**
… decomposition of the 4D **convolution** kernel **tensor** into a sum of a small number of rank-one …
Save Cite Cited by 956 Related articles All 4 versions

# Compression of fully-connected layers using tensors: idea

- Fully connected layers, say $1024 \times 1024$
- How we can apply tensors to it?
- Key idea: **tensorization**
- By using virtual dimensions, we can significantly reduce the number of parameters
- Inference speed is an issue, one approach is to develop **specialized hardware**

$$1024 = 2 \times 2 \times \ldots \times 2$$

$$A(i,j) \rightarrow A(i_1, \ldots, i_d; j_1, \ldots j_d) \rightarrow A(i_1, j_1; i_2 j_2; \ldots; i_d j_d)$$

The permutation of indices is important

**Tensorizing neural networks**

A **Novikov**, D Podoprikhin, A Osokin… - Advances in **neural** …, 2015 - proceedings.neurips.cc

… We will refer to a **neural network** with one or … **neural network** with 1024 hidden units and replace both fully-connected layers by the TT-layers. By setting all the TT-ranks in the **network** to …

☆ Save 99 Cite Cited by 881 Related articles All 14 versions

# Tensorized fully-connected layers

- We can not compress the pretrained models
- We need to **retrain the model from scratch**
- It is equivalent to the representation of a given layer in the form of d linear layers, where $d = \log N$

**Tensorizing neural networks**
A **Novikov**, D Podoprikhin, A Osokin… - Advances in **neural** …, 2015 - proceedings.neurips.cc
… We will refer to a **neural network** with one or … **neural network** with 1024 hidden units and replace both fully-connected layers by the TT-layers. By setting all the TT-ranks in the **network** to …
Save Cite Cited by 881 Related articles All 14 versions

# How can we optimize with low-rank tensor constraints

- Straight-forward option: implement the forward pass, use autograd. Works, but not optimally
- Use ADMM-methods (later)
- Use specialized Riemannian optimization

# Example: ADMM

$$\min_{\mathcal{W}} \ \ell(\mathcal{W}), \quad \text{s.t.} \ \mathcal{W} \in \mathcal{S},$$

$$g(\mathcal{W}) = \begin{cases} 0 & \mathcal{W} \in \mathcal{S}, \\ +\infty & \text{otherwise.} \end{cases}$$

$$\mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}, \mathcal{U}) = \ell(\mathcal{W}) + g(\mathcal{Z}) + \frac{\rho}{2}\|\mathcal{W} - \mathcal{Z} + \mathcal{U}\|_F^2 + \frac{\rho}{2}\|\mathcal{U}\|_F^2,$$

$$\mathcal{W}^{t+1} = \underset{\mathcal{W}}{\operatorname{argmin}} \ \mathcal{L}_\rho\left(\mathcal{W}, \mathcal{Z}^t, \mathcal{U}^t\right),$$
$$\mathcal{Z}^{t+1} = \underset{\mathcal{Z}}{\operatorname{argmin}} \ \mathcal{L}_\rho\left(\mathcal{W}^{t+1}, \mathcal{Z}, \mathcal{U}^t\right),$$
$$\mathcal{U}^{t+1} = \mathcal{U}^t + \mathcal{W}^{t+1} - \mathcal{Z}^{t+1},$$

Towards efficient **tensor** decomposition-based dnn model compression with optimization framework

M Yin, Y Sui, S Liao, B Yuan - Proceedings of the IEEE/CVF …, 2021 - openaccess.thecvf.com

… **tensor** decomposition, such as **tensor train** (TT) and **tensor** … Direction Method of Multipliers (**ADMM**). By formulating TT … with constraints on **tensor** ranks, we leverage **ADMM** technique to …

# Example: ADMM

**$\mathcal{Z}$-subproblem.** To solve $\mathcal{Z}$-subproblem (12), we first explicitly formulate it as follows:

$$\min_{\mathcal{Z}} \; g(\mathcal{Z}) + \frac{\rho}{2}\left\|\mathcal{W}^{t+1} - \mathcal{Z} + \mathcal{U}^t\right\|_F^2, \tag{17}$$

where the indicator function $g(\cdot)$ of the non-convex set $\mathcal{S}$ is non-differentiable. Then, according to [1], in this format updating $\mathcal{Z}$ can be performed as:

$$\mathcal{Z}^{t+1} = \mathbf{\Pi}_{\mathcal{S}}(\mathcal{W}^{t+1} + \mathcal{U}^t), \tag{18}$$

**$\mathcal{W}$-subproblem.** The $\mathcal{W}$-subproblem (11) can be reformulated as follows:

$$\min_{\mathcal{W}} \; \ell(\mathcal{W}) + \frac{\rho}{2}\left\|\mathcal{W} - \mathcal{Z}^t + \mathcal{U}^t\right\|_F^2, \tag{14}$$

where the first term is the loss function, e.g. cross-entropy loss in classification tasks, of the DNN model, and the second term is the $L_2$ regularization. This subproblem can be directly solved by SGD since both these two terms are differentiable. Correspondingly, the partial derivative of (14) with respect to $\mathcal{W}$ is calculated as

$$\frac{\partial \mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}^t, \mathcal{U}^t)}{\partial \mathcal{W}} = \frac{\partial \ell(\mathcal{W})}{\partial \mathcal{W}} + \rho(\mathcal{W} - \mathcal{Z}^t + \mathcal{U}^t). \tag{15}$$

And hence $\mathcal{W}$ can be updated by

$$\mathcal{W}^{t+1} = \mathcal{W}^t - \eta \frac{\partial \mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}^t, \mathcal{U}^t)}{\partial \mathcal{W}}, \tag{16}$$

Towards efficient **tensor** decomposition-based dnn model compression with optimization framework

M Yin, Y Sui, S Liao, B Yuan - Proceedings of the IEEE/CVF …, 2021 - openaccess.thecvf.com

… **tensor** decomposition, such as **tensor train** (TT) and **tensor** … Direction Method of Multipliers (**ADMM**). By formulating TT … with constraints on **tensor** ranks, we leverage **ADMM** technique to …

# Better scaling laws with structured layers

The most interesting case now are **transformer-based models**

The parameters are located in the **linear layers**

What if we parametrize those layers by fewer number of parameters?

We will get another **scaling laws:** loss vs number of parameters.

This has been studied recently!

**Compute Better Spent**: Replacing Dense Layers with Structured Matrices [PDF] arxiv.org
S Qiu, A Potapczynski, M Finzi, M Goldblum, AG **Wilson**
arXiv preprint arXiv:2406.06248, 2024 • arxiv.org

# Better scaling laws with structured layers

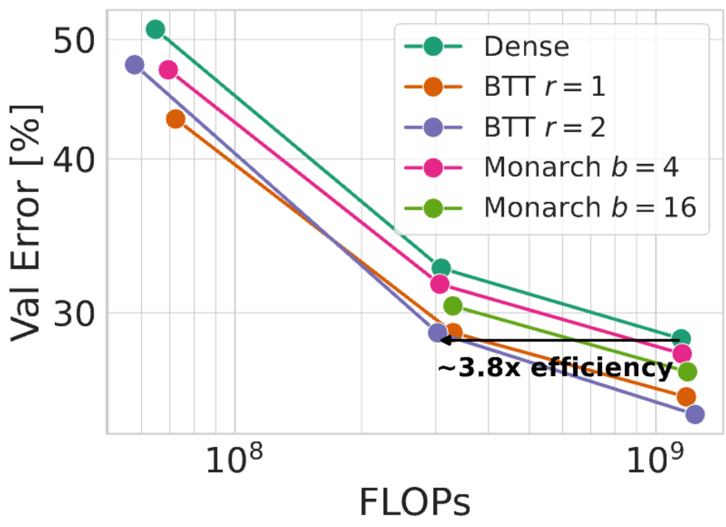
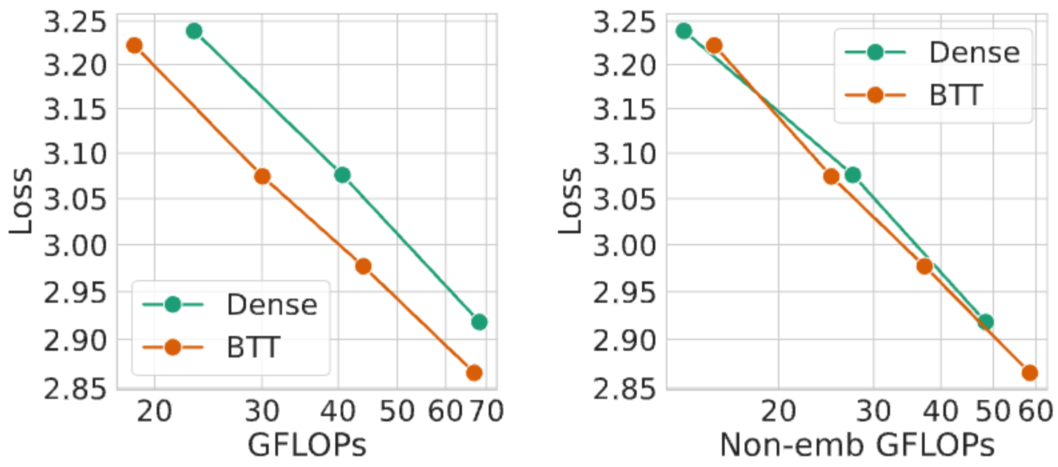



*Figure 8.* **ViTs trained on ImageNet with structured layers are more compute-efficient**. We use ViTs with patch size 32 trained for 300 epochs. BTT reaches the same performance of a dense ViT-S/32 with up to 3.8× fewer FLOPs.




(a) All compute (b) Non-embedding compute

*Figure 9.* **GPT-2 with all BTT layers is more compute-efficient.** (a) When including language modeling head compute, BTT is more efficient than dense. (b) When excluding language modeling head compute, BTT and dense perform similarly.

**Compute Better Spent**: Replacing Dense Layers with Structured Matrices [PDF] arxiv.org


S Qiu, A Potapczynski, M Finzi, M Goldblum, AG **Wilson**



arXiv preprint arXiv:2406.06248, 2024 • arxiv.org

# Block Tensor Train: class of structured matrices

| Structure | MVM FLOPs | # Params | Modeling assumptions | Example applications |
|---|---|---|---|---|
| Dense | $d^2$ | $d^2$ | General linear maps | MLPs, Transformers |
| Low-Rank | $2rd$ | $2rd$ | Compression | Bottleneck layers, Linear attention |
| Convolution | $pd$ | $p$ | Translation equivariance | Images, Time-series |
| Kronecker | $2d^{3/2}$ | $2d$ | Sets, Graphs, Grids | GPs, Deep Sets, Attention, GNNs |
| Monarch | $2d^2/b$ | $2d^2/b$ | Flexible | Compute-efficient linear layers |
| TT | $2rd^{3/2}$ | $2rd$ | Subsystems, Local interactions | Hidden Markov Models, Spin systems |
| BTT | $2rd^{3/2}$ | $2rd^{3/2}$ | Flexible | Compute-efficient linear layers |

*Table 1.* **Overview of the computational properties, modeling assumptions, and applications of structured matrices we consider.** Some structures require the same FLOPs as parameters for a matrix multiply, while others require more FLOPs. $d$ is the size of the matrix, $r$ is the rank in low-rank, TT, and BTT, $p$ is the kernel size in a convolution, and $b$ is the number of blocks in Monarch. We assume 2 cores each of size $\sqrt{d}$ for Kronecekr, TT and BTT.

**Compute Better Spent**: Replacing Dense Layers with Structured Matrices [PDF] arxiv.org
S Qiu, A Potapczynski, M Finzi, M Goldblum, AG **Wilson**
arXiv preprint arXiv:2406.06248, 2024 • arxiv.org

# Block Tensor Train: class of structured matrices

**Block Tensor-Train.** We propose a novel family of structured matrices called Block Tensor-Train (BTT) matrices, by removing the parameter-sharing along the block dimensions $\beta, \gamma$ in the TT structure. In the two core ($c = 2$) case, a BTT matrix of BTT-rank $r$ is defined by two parameter tensors $\mathbf{R} \in \mathbb{R}^{r \times \sqrt{d} \times \sqrt{d} \times \sqrt{d}}$ and $\mathbf{L} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d} \times \sqrt{d} \times r}$. Its MVM is given by

$$y_{\alpha\beta} = \sum_{\gamma\sigma} L_{\alpha\beta\gamma\sigma} \sum_{\delta} R_{\sigma\beta\gamma\delta} x_{\gamma\delta}. \qquad (2)$$

**Tensor-Train.** The Tensor-Train (TT) decomposition (Oseledets, 2011) specifies a set of $c$ cores $\mathbf{G}^{(i)} \in \mathbb{R}^{r_i \times m_i \times n_i \times r_{i-1}}$ for $i = 1, \ldots, c$ where $d = \prod_i m_i = \prod_i n_i$, $r_i \in \mathbb{N}$ and $r_0 = r_c = 1$. For ease of notation, we will focus on $c = 2$ with $m_1 = m_2 = n_1 = n_2 = \sqrt{d}, r_1 = r$, $\mathbf{G}^{(1)} = \mathbf{R} \in \mathbb{R}^{r \times \sqrt{d} \times \sqrt{d}}$, $\mathbf{G}^{(2)} = \mathbf{L} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d} \times r}$, though we present the general case in Appendix C. With the input and output as reshaped as $\sqrt{d} \times \sqrt{d}$ matrices, a TT matrix is equivalent to a sum over $r$ Kronecker products indexed by $\sigma = 1, \ldots, r$:

$$y_{\alpha\beta} = \sum_{\gamma\sigma} L_{\alpha\gamma\sigma} \sum_{\delta} R_{\sigma\beta\delta} x_{\gamma\delta}. \qquad (1)$$

**Compute Better Spent**: Replacing Dense Layers with Structured Matrices [PDF] arxiv.org
S Qiu, A Potapczynski, M Finzi, M Goldblum, AG **Wilson**
arXiv preprint arXiv:2406.06248, 2024 • arxiv.org

# Compression of embedding layers using tensor decomposition

- One of the recent promising directions is to compress **embedding layers**
- Embedding layer has the size $N_{voc} \times N_f$, we tensorize the 'id' dimension into a product of smaller numbers
- **Recent work:** reorder items for efficient compression, need specialized losses.

TensorGPT: Efficient Compression of the **Embedding Layer** in LLMs based on the Tensor-Train Decomposition
M Xu, YL Xu, DP Mandic - arXiv preprint arXiv:2307.00526, 2023 - arxiv.org
… Benefiting from the super-compression properties of Tensor Networks (TNs), we **tensorize** and decompose each token **embedding**, and then construct a highly efficient format of …
☆ Save 99 Cite Cited by 1 All 2 versions

A **tensorized** transformer for language modeling
X Ma, P Zhang, S Zhang, N Duan… - Advances in neural …, 2019 - proceedings.neurips.cc
… **Tensorized embedding** (TE) [18] uses the tensor-train [25] to compress the **embedding layers** in Transformer-XL [7], but has not compressed the attention **layer**. Recently, Block-Term …
☆ Save 99 Cite Cited by 127 Related articles All 12 versions

[PDF] **Tensorized embedding layers** for efficient model compression
V Khrulkov, O Hrinchuk, L Mirvakhabova… - arXiv preprint arXiv …, 2019 - researchgate.net
… **embedding layers**, we can greatly compress the entire model by compressing these **layers**, … Our goal is to replace the standard **embedding layer** specified by an **embedding** matrix with …
☆ Save 99 Cite Cited by 43 Related articles

# Compression of embedding layers using tensor decomposition

- Option 1: Train embedding layers from scratch
- Option 2: Compress a pretrained embedding layer, for example, for FaceID

# Post-compression of embedding layers

- We are given a large matrix of size $N_{id} \times N_f$
- The ordering of indices does not matter!
- We look for a TT-matrix such that $TT \approx PA$, where P is a permutation matrix
- The naive choice of the loss is the Wasserstein loss.
- More interesting is to have a **hierarchical clusterization**

# Post-compression of embedding layers

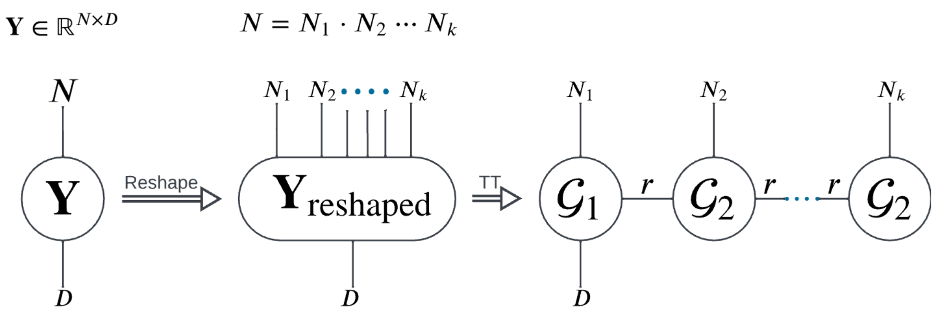



Figure 1: Diagram of the TT point cloud in Penrose graphical notation. Each tensor is depicted as a vertex, and each vertex has as many edges as the dimensionality of the corresponding tensor. Two tensors are connected with a common edge if these two tensors are contracted along the corresponding dimension.

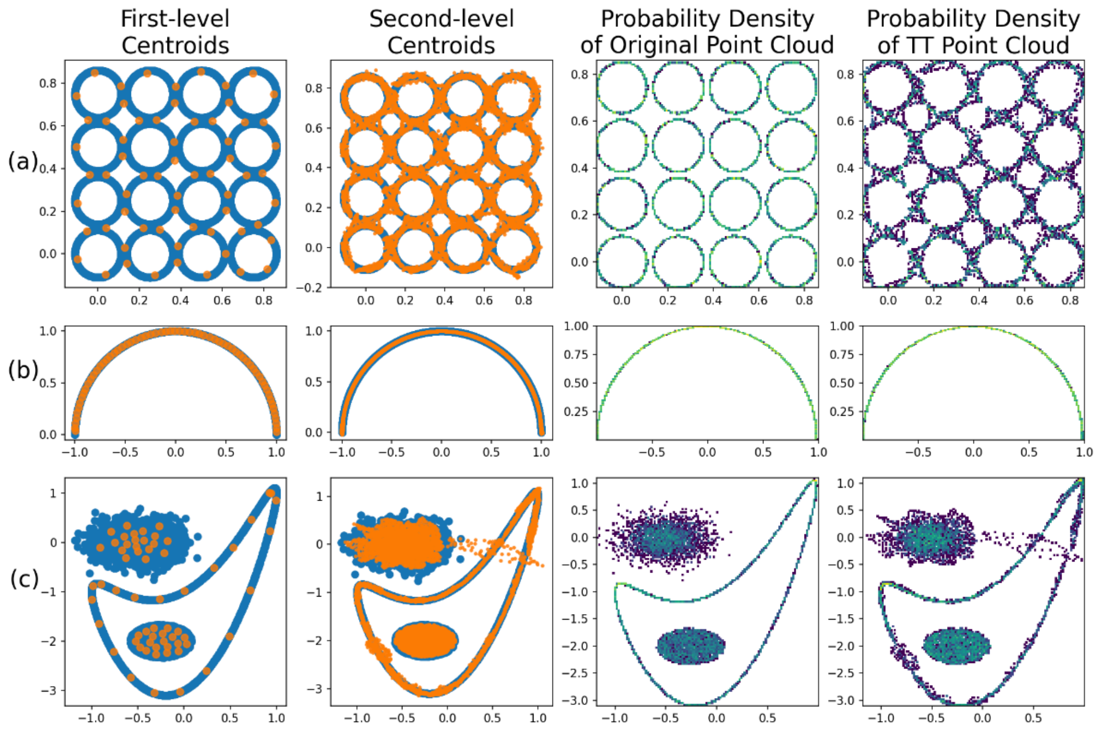



Figure 2: Three toy point clouds (blue points) consisting of 8192 vectors each, and its compressed TT-point cloud approximation (orange points).



AIRI

# Tensor-based models for machine learning: key idea

- Can we build tensor representations into the ML pipelines?
- Yes, we can but not without difficulties
- First proposed in Exponential Machines paper
- Pioneering generalization by N. Cohen
- Our followup on connection between recurrent neural networks and tensor train decomposition

$x = (x_1, \ldots, x_d)$, i.e. patches

Rank-1 feature map:

$$\Psi(x) = f_1(x_1) \otimes \ldots \otimes f_d(x_d)$$

Linear model in this space:

$$l(x) = \langle W, \Phi \rangle$$

Put low-rank constraints on W!

On the expressive power of deep learning: A tensor analysis
N **Cohen**, O Sharir, A **Shashua** - Conference on learning …, 2016 - proceedings.mlr.press
It has long been conjectured that hypotheses spaces suitable for data that is compositional in nature, such as text or images, may be more efficiently represented with deep hierarchical …
Save Cite Cited by 517 Related articles All 12 versions

**Expressive power** of **recurrent neural** networks
V Khrulkov, A Novikov, I Oseledets - arXiv preprint arXiv:1711.00811, 2017 - arxiv.org
… shallow network) for a class of **recurrent neural** networks – ones that correspond to the Tensor … compare **expressive powers** of the HT- and TT-Networks. We also implement the **recurrent** …
Save Cite Cited by 109 Related articles All 5 versions

**Exponential machines**
A Novikov, M Trofimov, I Oseledets - arXiv preprint arXiv:1605.03795, 2016 - arxiv.org
… the performance of **machine** learning solutions in many … **Exponential Machines** (ExM), a predictor that models all interactions of every order. The key idea is to represent an **exponentially** …
Save Cite Cited by 103 Related articles All 4 versions

# Tensor-train density estimation

- One example: density estimation with tensors
- One can use simple losses, because integration is easy
- Works fast for tabular data

$$\mathscr{L}(p, q_\theta) = \int \left(p(\boldsymbol{x}) - q_\theta(\boldsymbol{x})\right)^2 d = \int q_\theta(\boldsymbol{x})^2 dx - 2\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} q_\theta(\boldsymbol{x}) + \text{const}$$

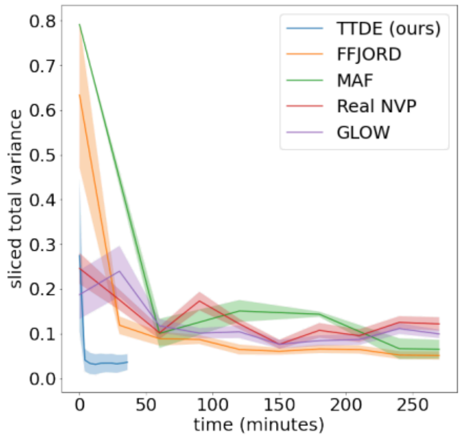



Figure 4: Dependence of the sliced total variation w.r.t. the training time for models trained on 6-dimensional UCI POWER dataset.

# Tensor-based optimization: Quantum-inspired algorithms

Recent results focus on connecting tensor approximation with optimization

The idea is very simple: approximate the function by sampling with a tensor decomposition, hopefully get the maximum element

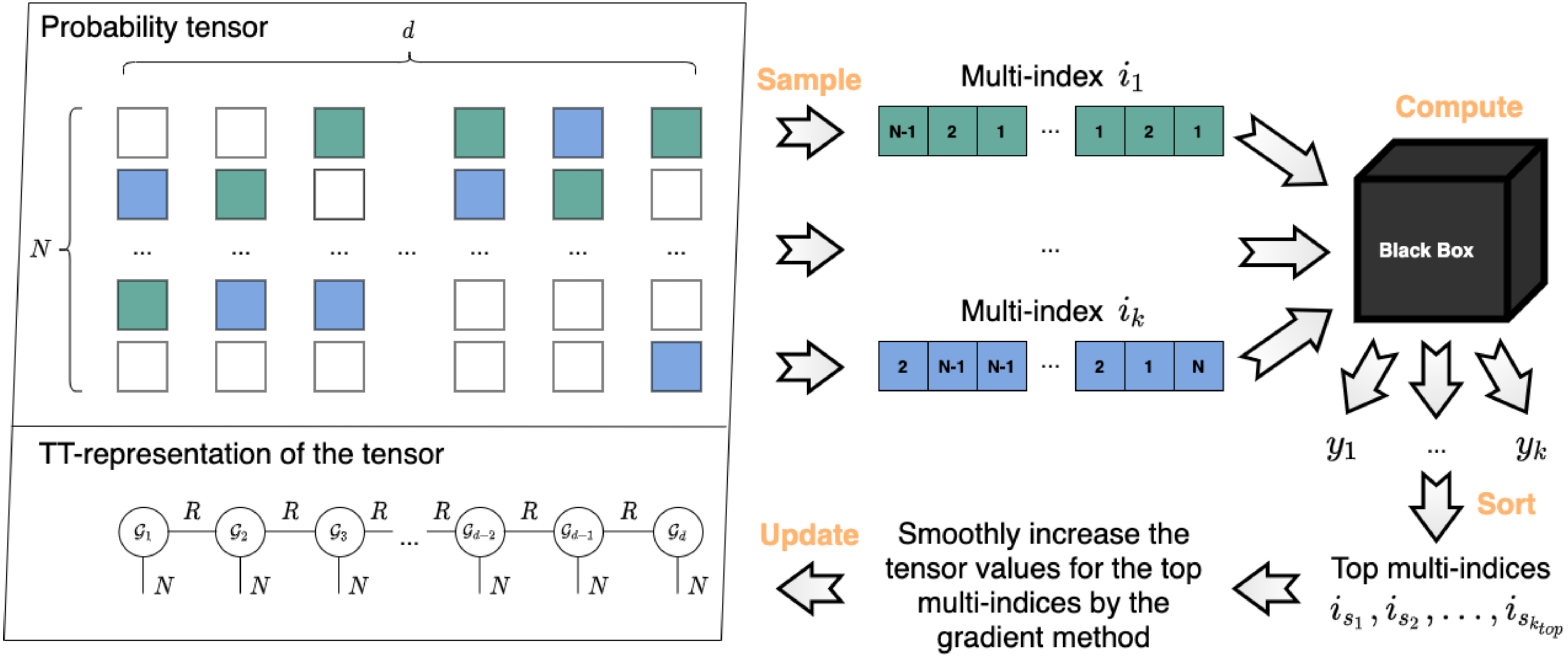
PROTES: Probabilistic optimization with tensor sampling.

Idea: Sample candidates from the probability distribution



AIRI

# Our approach: PROTES
# **Pro**babilistic **O**ptimization **with TE**nsor **S**ampling



# Our approach: PROTES
# **Pro**babilistic **O**ptimization **with TE**nsor **S**ampling

Comparison with Nevergrad (Meta) and other approaches

Table 1: Minimization result for all selected benchmarks (P-01 – P-20). We report the values obtained by the proposed method PROTES and by all considered baselines (BS1 – BS7). For each benchmark, the best result is highlighted in bold. The last row presents the number of best results for each baseline.

| | | OUR | BS-1 | BS-2 | BS-3 | BS-4 | BS-5 | BS-6 | BS-7 |
|---|---|---|---|---|---|---|---|---|---|
| ANALYTIC FUNCTIONS | P-01 | **9.1E+00** | **9.1E+00** | **9.1E+00** | **9.1E+00** | **9.1E+00** | 2.0E+01 | **9.1E+00** | **9.1E+00** |
| | P-02 | **1.7E+00** | **1.7E+00** | **1.7E+00** | 2.7E+00 | **1.7E+00** | 5.4E+00 | 2.4E+00 | 1.9E+00 |
| | P-03 | **-9.8E-01** | **-9.8E-01** | **-9.8E-01** | **-9.8E-01** | **-9.8E-01** | -6.8E-01 | **-9.8E-01** | **-9.8E-01** |
| | P-04 | **4.5E+00** | **4.5E+00** | **4.5E+00** | **4.5E+00** | **4.5E+00** | 7.4E+01 | **4.5E+00** | **4.5E+00** |
| | P-05 | **-5.4E+00** | **-5.4E+00** | **-5.4E+00** | -3.9E+00 | -5.0E+00 | -1.9E+00 | -1.6E+00 | -5.3E+00 |
| | P-06 | **1.6E-01** | **1.6E-01** | **1.6E-01** | **1.6E-01** | **1.6E-01** | 1.9E-01 | 4.4E-01 | **1.6E-01** |
| | P-07 | **2.3E+07** | **2.3E+07** | **2.3E+07** | 4.8E+07 | 2.9E+07 | 9.6E+09 | 1.4E+11 | **2.3E+07** |
| | P-08 | **2.7E+01** | **2.7E+01** | **2.7E+01** | 6.7E+01 | **2.7E+01** | 8.3E+01 | 1.0E+02 | **2.7E+01** |
| | P-09 | **1.2E+00** | **1.2E+00** | **1.2E+00** | 1.4E+00 | **1.2E+00** | 2.5E+00 | 1.7E+00 | **1.2E+00** |
| | P-10 | **4.4E+02** | **4.4E+02** | **4.4E+02** | 8.3E+02 | 7.6E+02 | 1.7E+03 | 2.8E+03 | **4.4E+02** |
| QUBO | P-11 | **-3.6E+02** | -3.5E+02 | -3.4E+02 | -3.2E+02 | -3.4E+02 | -3.2E+02 | 0.0E+00 | **-3.6E+02** |
| | P-12 | **-5.9E+03** | **-5.9E+03** | **-5.9E+03** | -5.2E+03 | -5.7E+03 | -5.4E+03 | **-5.9E+03** | **-5.9E+03** |
| | P-13 | **-3.8E+00** | -3.7E+00 | -3.4E+00 | -2.8E+00 | 1.1E+01 | 7.4E+02 | -1.2E+00 | **-3.8E+00** |
| | P-14 | **-3.1E+03** | -2.9E+03 | -2.2E+03 | -2.5E+03 | -2.9E+03 | -2.6E+03 | -2.9E+03 | -3.0E+03 |
| CONTROL | P-15 | **6.8E-03** | 8.4E-03 | 5.5E-01 | 1.4E-02 | 9.9E-03 | 1.7E-02 | 1.7E-01 | 7.9E-03 |
| | P-16 | **1.4E-02** | 3.0E-02 | 2.3E-01 | 4.3E-02 | 1.7E-02 | 4.9E-02 | 2.5E-01 | 1.5E-02 |
| | P-17 | **3.0E-02** | 3.4E-01 | 2.1E+00 | 5.0E-02 | 3.2E-02 | 1.1E-01 | 1.4E+00 | 3.6E-02 |
| CONTROL +CONSTR. | P-18 | **1.3E-02** | 1.5E-02 | FAIL | 4.8E-02 | 9.1E-02 | FAIL | 2.5E-01 | 5.6E-02 |
| | P-19 | **1.7E-02** | 1.6E+00 | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL |
| | P-20 | **4.7E-02** | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL | FAIL |
| WINS | | 20 | 11 | 11 | 4 | 7 | 0 | 4 | 11 |

**PROTES**: Probabilistic Optimization with Tensor Sampling
…, A Chertkov, G Ryzhakov, I **Oseledets** - arXiv preprint arXiv …, 2023 - arxiv.org
We develop new method **PROTES** for optimization of the multidimensional arrays and discretized multivariable functions, which is based on a probabilistic sampling from a probability …
☆ Save 99 Cite Cited by 1 All 2 versions

# Kashin decomposition (UAI 2024 paper)

Fundamental result by Boris Kashin:

Every vector $x \in \mathbb{R}^d$ can be represented as

$x = u + Qv$ where $Q$ is a (random) orthogonal matrix and

$$\|u\|_\infty \leq \frac{c}{\sqrt{N}}, \quad \|v\|_\infty \leq \frac{c}{\sqrt{N}}$$

Bounded infinity norm = good quantization!



AIRI

# Kashin decomposition algorithm

**Algorithm 1:** Kashin Decomposition Algorithm

**Input:** Vector $x \in \mathbb{R}^n$, Orthogonal matrix $Q$, Tolerance $\varepsilon > 0$

**Output:** Vectors $u, \hat{v} \in \mathbb{R}^n$ such that $x \approx u + \hat{v} = u + Qv$, and both $u$ and $v$ have small infinity norms.

Define projection $\pi_x(y) \coloneqq \frac{x^\top y}{\|y\|_2^2} \cdot y$

Initialize $u \leftarrow 0^n, \hat{v} \leftarrow 0^n$

**while** $\|x - u - \hat{v}\| \geq \varepsilon$ **do**

  **if** $\|x\|_1 > \|Q^T x\|_1$ **then**

    $\pi \leftarrow \pi_x(\mathsf{Sign}(x))$

    $u \leftarrow u + \pi$

  **else**

    $\pi \leftarrow \pi_x(Q\mathsf{Sign}(Q^T x))$

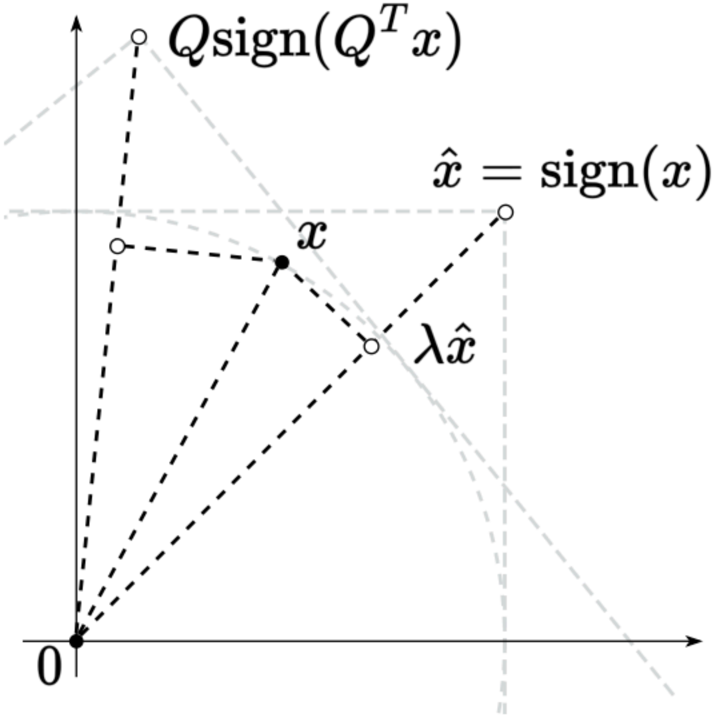
    $\hat{v} \leftarrow \hat{v} + \pi$

  **end**

  $x \leftarrow x - \pi$

**end**

**return** $x, u, \hat{v}$



# Kashin decomposition algorithm: how the elements look like

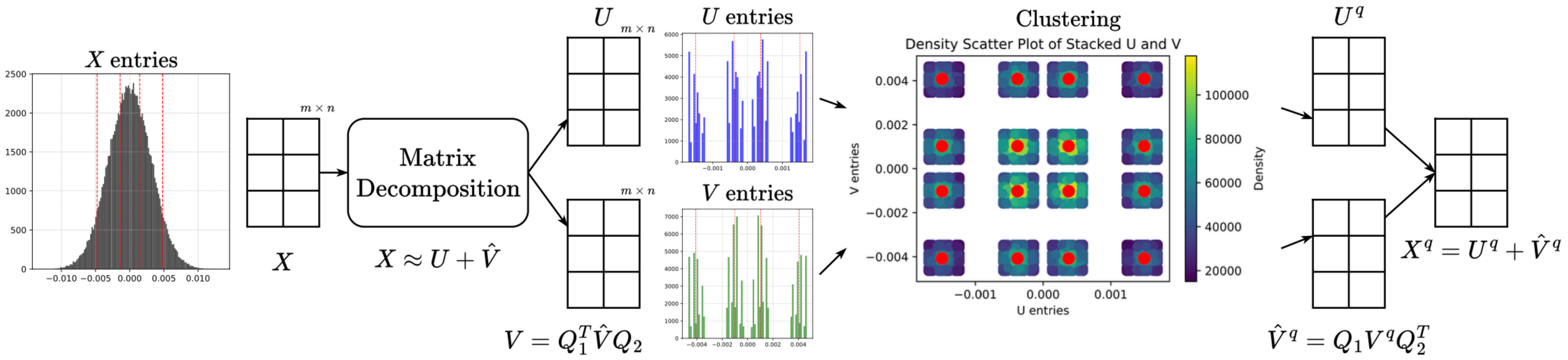



Figure 1: We solve the post-training quantization problem, i.e., replacing the original weights of an LLM with a low-bit representation. We split the matrix into two factors with Kashin decomposition algorithm, whose values are replaced with the closest centroids, so $X^q = U^q + \hat{V}^q$. The number of centroids determines the compression level (4-bit).

# Kashin decomposition algorithm: results

| Quantization | COLA | SST-2 | QQP | QNLI | MNLI | RTE | STS-B | MRPC | WNLI |
|---|---|---|---|---|---|---|---|---|---|
| **roberta-base** | | | | | | | | | |
| FP32 | 59.06 | 93.8 | 91.24/88.36 | 92.62 | 88.1/87.43 | 67.87 | 89.65/89.49 | 87.75/91.2 | 56.34 |
| Uniform 4bit | 0.0 | 49.08 | 36.82/53.82 | 49.46 | 31.82/31.82 | 52.71 | 10.08/9.37 | 68.38/81.22 | 56.34 |
| Kmeans 4bit | 46.97 | **92.77** | 88.77/**87.39** | 89.27 | 83.98/82.85 | 55.23 | 79.65/80.47 | 71.32/75.77 | 56.34 |
| Kashin 4bit (ours) | **52.09** | 90.37 | **89.53**/86.73 | **91.14** | **86.41/85.51** | **60.28** | **87.29/87.26** | **83.08/87.10** | 56.34 |
| **bert-base** | | | | | | | | | |
| FP32 | 59.31 | 91.74 | 90.66/87.39 | 90.74 | 83.96/84.24 | 64.98 | 88.94/88.77 | 84.31/88.81 | 42.25 |
| Uniform 4bit | 1.24 | 49.66 | 38.09/52.75 | 49.22 | 32.24/33.34 | 50.18 | -0.21/-0.25 | 63.97/76.02 | 49.3 |
| Kmeans 4bit | 54.43 | 91.51 | 88.87/85.33 | 88.01 | 78.63/78.76 | 55.23 | 85.06/85.0 | 34.55/8.87 | **54.92** |
| Kashin 4bit (ours) | **59.65** | **91.63** | **90.28/87.33** | **90.06** | **83.88/84.01** | **63.53** | **88.76/88.56** | **84.80/89.31** | 42.25 |

Table 2: Results of GLUE benchmark for Bert and RoBerta models, where we quantized linear layers in transformer blocks with three quantization methods: uniform, kmeans, and Kashin quantization.

AIRI


# Artificial Intelligence Research Institute

airi.net

- airi_research_institute
- AIRI Institute
- AIRI Institute
- AIRI_inst
- artificial-intelligence-research-institute