Faster & Deterministic FPT Algorithm for Worst-Case Tensor Decomposition

Vishwas Bhargava[1] Devansh Shringi[2]

[1]California Institute of Technology

[2]University of Toronto

ICALP, July 2025

Tensors

- **Tensor:** Higher Dimension versions of Matrices

$$\mathcal{T} = (\alpha_{j_1,j_2,\ldots,j_d}) \in \mathbb{F}^{n_1 \times \cdots \times n_d}$$

$\mathcal{T}$ is a d-dimensional tensor with shape $[n_1] \times \ldots \times [n_d]$. $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$

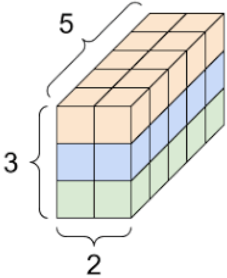


Figure: A 3-dimensional $[3] \times [5] \times [2]$ shaped Tensor

UNIVERSITY OF TORONTO

Tensor Rank

- Rank of a Tensor (CP rank)

Tensor rank

Minimum k s.t. $\mathcal{T}$ can be written as sum of k rank-1 tensors

- **Tensor Decomposition:** Computing the k rank-1 tensors

$$\mathcal{T} = \sum_{i=1}^{k} v_{i1} \otimes \cdots \otimes v_{id}$$

where $v_{ij} \in \mathbb{F}^{n_j}$, $\otimes$ is the Kronecker/Outer Product

Tensor Decomposition: Applications

An algorithmic primitive with many applications

- **Complexity** (Matrix Multiplication) and **Combinatorics** (Capset, Sunflower)
- **Machine Learning** (Learning Mixture of Gaussians, Dictionary Learning, Topic Modeling, etc.)
- **Statistics** (Cumulants, Blind source separation, etc.)
- **Signal processing** (Independent component analysis, Community detection, Multi-reference alignment, etc.)
- Phylogenetic reconstruction, Quantum Information Theory, Fluorescence spectroscopy.

(see [Lan12])

UNIVERSITY OF TORONTO

Tensor Decomposition: Complexity

- [Hås90] showed finding Tensor rank is NP-hard for $d \geq 3$.
- [SS16] hardness related to solving a system of polynomial equations.

Tensor Decomposition: Complexity

- [Hås90] showed finding Tensor rank is NP-hard for $d \geq 3$.
- [SS16] hardness related to solving a system of polynomial equations.
- Studied in various settings such as finding decomposition for Random Tensors, Generic tensors, and Worst-case analysis.
- We are interested in the setting of worst-case tensor decomposition where $d \approx \mathrm{poly}(n), k \approx \log^{1/c} n$ for some constant c.
- Our Goal is to come up with **Deterministic** Tensor decomposition algorithms that run in time $2^{\mathrm{poly}(k)} \cdot \mathrm{poly}(n, d)$.

UNIVERSITY OF TORONTO

Reconstruction of Arithmetic Circuits

Arithmetic Circuits

- Circuit computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$.
- Gates are $+, \times$. Leaves have $\{x_1, \ldots, x_n, \mathbb{F}\}$.
- Edges with labels from $\mathbb{F}$ (1 by default).

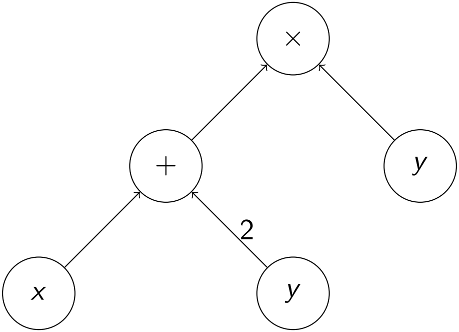


Figure: Circuit computing $xy + 2y^2$

Reconstruction of Arithmetic Circuits

Reconstruction

Let f be a polynomial computed by a circuit C from class $\mathcal{C}$. Given Black-box access to evaluations of a f, efficiently output a circuit computing f.

- **Proper learning:** The output circuit should also be from the class $\mathcal{C}$.

UNIVERSITY OF TORONTO

Reconstruction of Arithmetic Circuits

Reconstruction

Let f be a polynomial computed by a circuit C from class $\mathcal{C}$. Given Black-box access to evaluations of a f, efficiently output a circuit computing f.

- **Proper learning:** The output circuit should also be from the class $\mathcal{C}$.
- **Depth-2:** $\Sigma\Pi$ Interoplation [BOT88, KS01], $\Pi\Sigma$ Factoring [Kal87]
- Polytime reconstruction for Depth-3 Arithmetic circuits $\Sigma\Pi\Sigma$ give subexponential time reconstruction for general circuits [GKKS13].
- Polytime PAC-Learning for depth-3 circuits give quantum polytime SVP algorithms [KS09b].

Set-Multilinear Polynomials

- **Multilinear:** In each monomial, the degree of each variable is 1
- **Set-Multilinear:** Variables partitioned into $X = \sqcup_{j \in [d]} X_j$, each monomial is of the form $x_{i_1} x_{i_2} \dots x_{i_d}$ where $x_{i_j} \in X_j$.

Set-Multilinear Polynomials

- **Multilinear:** In each monomial, the degree of each variable is 1
- **Set-Multilinear:** Variables partitioned into $X = \sqcup_{j\in[d]} X_j$, each monomial is of the form $x_{i_1} x_{i_2} \ldots x_{i_d}$ where $x_{i_j} \in X_j$.
- A circuit is multilinear/set-multilinear if the polynomial computed at every gate is multilinear/set-multilinear.

Depth-3 Set-Multilinear circuits $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$

$$C(X) = \sum_{i=1}^{k} \prod_{j=1}^{d} \ell_{i,j}(X_j)$$

UNIVERSITY OF TORONTO

Equivalence of the Two problems

- Tensor $\mathcal{T}$ d-dimensional with shape $[n_1] \times \ldots \times [n_d]$

$$\mathcal{T} = (\alpha_{j_1,j_2,\ldots,j_d}) \in \mathbb{F}^{n_1 \times \cdots \times n_d}$$

- $\mathcal{T}$ has decomposition

$$\mathcal{T} = \sum_{i=1}^{k} v(\ell_{i,1}) \otimes \cdots \otimes v(\ell_{i,d})$$

where $v(\ell_{i,j})$ is $\ell_{i,j}$ as vector in $\mathbb{F}^{n_j}$.

- Set-Multilinear Polynomial $f_{\mathcal{T}}$ over variables $X = \sqcup_i X_i$, $X_i = \{x_{i,1}, \ldots, x_{i,n_i}\}$

$$f_{\mathcal{T}} := \sum_{\bar{j} \in \mathcal{T}} \alpha_{j_1,j_2,\ldots,j_d} x_{1,j_1} x_{2,j_2} \ldots x_{d,j_d}$$

- $C(X)$ is a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit computing $f_{\mathcal{T}}$

$$f_{\mathcal{T}} = C(X) := \sum_{i=1}^{k} \prod_{j=1}^{d} \ell_{i,j}(X_j)$$

UNIVERSITY OF TORONTO

Past Work

Past Work

Results	Algorithm Type	Running Time	Works for
[Shp07]	Randomized	$\mathrm{poly}(n, d, \lvert\mathbb{F}\rvert)$	*Multilinear* $\Sigma\Pi\Sigma(2)$
[KS09a]	Deterministic	$\mathrm{poly}(n, d^{k^2}, \lvert\mathbb{F}\rvert)$	*Multilinear* $\Sigma\Pi\Sigma(k)$
[BSV21]	Deterministic	$\mathrm{poly}(d^{k^3}, k^{k^{k^{10}}}, n)$	Both (Multi./SM)
[PSV24]	Randomized	$k^{k^{k^{\mathcal{O}(k)}}} \cdot \mathrm{poly}(n, d)$	Both (Multi./SM)
[BS25]	Deterministic	$2^{\mathrm{poly}(k)} \cdot \mathrm{poly}(n, d)$	Set-Multilinear only

Table: Comparison of our results to previous works.

Past Work

Results	Algorithm Type	Running Time	Works for
[Shp07]	Randomized	$\mathrm{poly}(n, d, \|\mathbb{F}\|)$	*Multilinear* $\Sigma\Pi\Sigma(2)$
[KS09a]	Deterministic	$\mathrm{poly}(n, d^{k^2}, \|\mathbb{F}\|)$	*Multilinear* $\Sigma\Pi\Sigma(k)$
[BSV21]	Deterministic	$\mathrm{poly}(d^{k^3}, k^{k^{k^{10}}}, n)$	Both (Multi./SM)
[PSV24]	Randomized	$k^{k^{k^{\mathcal{O}(k)}}} \cdot \mathrm{poly}(n, d)$	Both (Multi./SM)
[BS25]	Deterministic	$2^{\mathrm{poly}(k)} \cdot \mathrm{poly}(n, d)$	Set-Multilinear only

Table: Comparison of our results to previous works.

NP-hardness and Exponential time Hypothesis $\implies 2^{\Omega(k)} \cdot \mathrm{poly}(n, d)$ running time for tensor decomposition.

Our Results

Our Results

Theorem (Learning $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuits)

Given blackbox access to degree d, n variate polynomial f computable by a set-multilinear $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit C with top fan-in k over $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, then there exists a ***deterministic*** *algorithm that outputs a set-multilinear $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit over $\mathbb{F}$ with top fan-in k computing f in time $F(k, n, d) = 2^{\mathrm{poly}(k)} \cdot \mathrm{poly}(n, d)$.*

Difficult to improve over $2^{\mathrm{poly}(k)}$ using approaches that use solving system of polynomial equations.

UNIVERSITY OF TORONTO

Our Results

Corollary (Decomposing rank-k tensors)

*Let $\mathcal{T} \in \mathbb{F}^{n_1 \times \cdots \times n_d}$ be a d-dimensional tensor of rank at most k with $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$. Let $n = \sum_{i=1}^{d} n_i$. Given black-box access to measurements of $\mathcal{T}$ (equivalently to evaluations of $f_{\mathcal{T}}$), there exists a **deterministic** $\mathrm{poly}(2^{\mathrm{poly}(k)}, d, n)$ time algorithm for computing a decomposition of $\mathcal{T}$ as a sum of at most k rank 1 tensors.*

Our Results: Finite Fields

Theorem (Learning $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuits over $\mathbb{F}_q$)

Given blackbox access to degree d, n variate polynomial f computable by a set-multilinear $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit C with top fan-in k over $\mathbb{F} = \mathbb{F}_q$, then there exists a ***randomized*** *algorithm that outputs a set-multilinear $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit over $\mathbb{F}$ with top fan-in k computing f in time $F(k, n, d) = 2^{2^{\mathrm{poly}(k)}} \cdot \mathrm{poly}(n, d)$.*

Time Blow-up and Randomization are both from procedures for solving systems of polynomial equations.

Proof Idea: Required Tools

Polynomial Identity Testing

- For $\Sigma\Pi\Sigma$ circuit $C = \sum_{i=1}^{k} T_i$

$$\Delta(T_1, T_2) = \mathrm{rank}(\mathrm{sim}(T_1 + T_2)) = \dim(\{\ell_{i,j} : \ell_{i,j} \notin \gcd(T_1, T_2)\})$$

- Let $C = \sum_{i=1}^{k} T_i$ be $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ computing 0.

$$\forall i \neq j \in [k],\ \Delta(T_i, T_j) \leq k - 2$$

Polynomial Identity Testing

- For $\Sigma\Pi\Sigma$ circuit $C = \sum_{i=1}^{k} T_i$

$$\Delta(T_1, T_2) = \mathrm{rank}(\mathrm{sim}(T_1 + T_2)) = \dim(\{\ell_{i,j} : \ell_{i,j} \notin \mathrm{gcd}(T_1, T_2)\})$$

- Let $C = \sum_{i=1}^{k} T_i$ be $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ computing 0.

$$\forall i \neq j \in [k],\ \Delta(T_i, T_j) \leq k - 2$$

- [GG20] Deterministic PIT for $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ in time $2^{\mathcal{O}(\log^2 k)} \cdot \mathrm{poly}(n)$
- [SV10] Assume $\mathrm{gcd}(T_1, \ldots, T_k) = 1$.

UNIVERSITY OF TORONTO

Solving System of Polynomial

Theorem

Let $f_1, f_2, \ldots f_m \in \mathbb{F}[x_1, \ldots, x_n]$ be n-variate polynomials of degree at most d. Then, the complexity of finding a single solution to the system $f_1(x) = 0, \ldots, f_m(x) = 0$ (if one exists) over various fields is as follows:

1. *[GV88] For $\mathbb{F} = \mathbb{R}$, deterministic $\mathrm{Sys}_{\mathbb{F}}(n, m, d) = \mathrm{poly}((md)^{n^2})$ time.*
2. *[Ier89] For $\mathbb{F} = \mathbb{C}$ (or any algebraically closed field) deterministic $\mathrm{Sys}_{\mathbb{F}}(n, m, d) = (mn)^{O(n)} \cdot d^{O(n^2)}$ time.*
3. *For all fields $\mathbb{F}$, the $\mathrm{Sys}_{\mathbb{F}}(n, m, d) = \mathrm{poly}((nmd)^{3^n})$.*

Proof Idea: Techniques from BSV21

Low-Degree Reconstruction

Lemma ([BSV21])

Given black-box access to a degree d polynomial $f \in \mathbb{F}[X]$ such that f is computable by a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit C_f over the field $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, there is a deterministic $2^{\mathrm{poly}(k,d)} \cdot \mathrm{poly}(n,d)$ time algorithm that outputs a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit computing f.

UNIVERSITY OF TORONTO

Learning Almost Circuit

Lemma (Observed from [BSV21])

Given black-box access to an $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ *circuit* $C = T_1 + T_2 + \ldots + T_k$ *computing* f*, there exists an algorithm that runs in time* $2^{\mathrm{poly}(k)} \cdot \mathrm{poly}(n, d)$ *and outputs a* $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ *circuit* C' *such that* $C' = T'_1 + T'_2 + \ldots + T'_k$ *has the property that* $\forall\, i \in [k]\ \Delta(T_i, T'_i) < 2k$*.*

Bottleneck: Brute-force over $\binom{d}{k^3}$ choices of variable parts in [BSV21] to go from C' to C.

Proof Idea: New Ideas

Learning with Assumption

- $\mathrm{Consistent} - \mathrm{VarPart}(C, C') := \{j \in [d] | \forall i \in [k] \ell_{i,j} \in T_i \cap T'_i\}$
- $|\mathrm{Consistent} - \mathrm{VarPart}(C, C')| \geq d - k^2$.
- **Assumption:** $\exists j \in \mathrm{Consistent} - \mathrm{VarPart}(C, C')$ such that

$$\ell_{1,j} \notin \mathrm{span}(\ell_{2,j}, \ldots, \ell_{k,j})$$

Learning with Assumption

- $\mathrm{Consistent-VarPart}(C, C') := \{j \in [d] | \forall i \in [k] \ell_{i,j} \in T_i \cap T_i'\}$
- $|\mathrm{Consistent-VarPart}(C, C')| \geq d - k^2$.
- **Assumption:** $\exists j \in \mathrm{Consistent-VarPart}(C, C')$ such that
$$\ell_{1,j} \notin \mathrm{span}(\ell_{2,j}, \ldots, \ell_{k,j})$$
- Set $X_j = \bar{\alpha}_j$ such that $T_2, \ldots, T_k$ vanish, but T_1 doesn't. Learn $T_1(X_j = \bar{\alpha}_j)$.
- Recover $T_1 := T_1(X_j = \bar{\alpha}_j) \cdot \frac{\ell_{1,j}}{\ell_{1,j}(\bar{\alpha}_j)}$.
- Reconstruct $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k-1)$ circuit $C - T_1$.

Assumption is not True

- Iterate over variable parts in $\mathrm{Consistent} - \mathrm{VarPart}(C, C')$ and Decrease fan-in by 1.
- Go until only T_1 is alive.

Assumption is not True

- Iterate over variable parts in $\mathrm{Consistent} - \mathrm{VarPart}(C, C')$ and Decrease fan-in by 1.
- Go until only T_1 is alive.
- What if T_1, T_i are same on all variable parts in $\mathrm{Consistent} - \mathrm{VarPart}(C, C')$? i.e. C' cannot help us differentiate between T_1, T_i.

Clustering

- Cluster gates close to T_1 together in set $A \subseteq [k]$.
- New Goal, learn C_A.

Lemma

$\exists$ *clustering* $A \subseteq [k]$ *s.t.*

$$\forall i \notin A,\ \Delta_A(C_A, T_i) \geq k^2 + k$$

and

$$\Delta(C_A) = \mathrm{rank}(\mathrm{sim}(\sum_{i \in A} T_i)) \leq k^4 + k^3$$

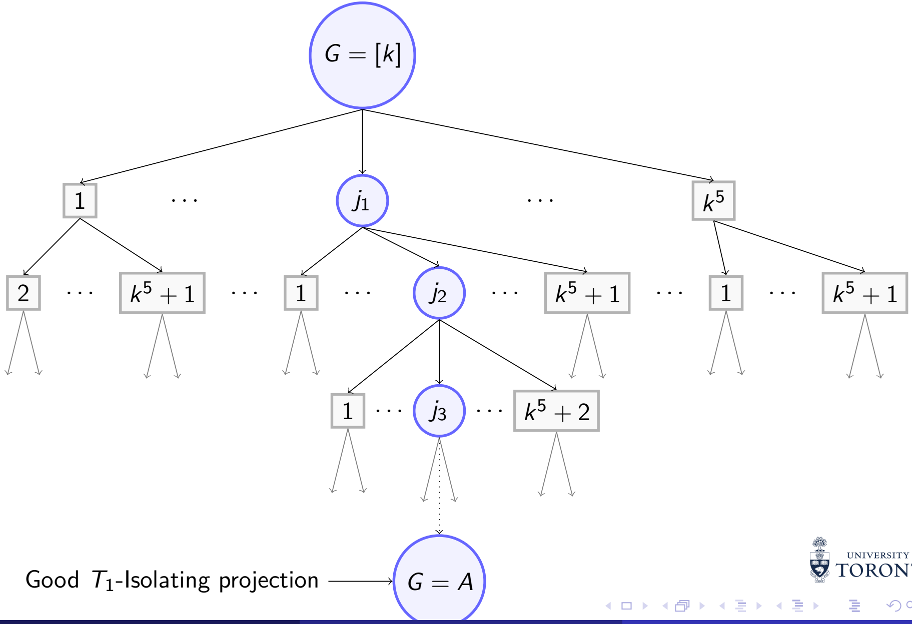
Isolating Cluster

- $\mathrm{Good-Proj} = \mathrm{Consistent-VarPart}(C, C') \cap \mathrm{Supp}(\mathrm{Lin}(C_A))$.
- $|\mathrm{Good-Proj}| \geq d - 2k^4 - k^3$. Use $\mathrm{Good-Proj}$ instead of $\mathrm{Consistent-VarPart}(C, C')$.

Isolating Cluster

- $\mathrm{Good-Proj} = \mathrm{Consistent-VarPart}(C, C') \cap \mathrm{Supp}(\mathrm{Lin}(C_A))$.
- $|\mathrm{Good-Proj}| \geq d - 2k^4 - k^3$. Use $\mathrm{Good-Proj}$ instead of $\mathrm{Consistent-VarPart}(C, C')$.
- Pick $k^5 + 1$ var. parts in $[d]$, at least 1 in $\mathrm{Good-Proj}$
- Set $X_j = \bar{\alpha}_j$ s.t. C_A survives, some $T_i \notin A$ is set to 0.

Isolating Cluster



Learning Cluster

- Cluster is isolated after at most k settings $(\sigma := (X_{j_1} = \bar{\alpha}_{j_1}), \ldots)$
- Factor and Reconstruct $\mathrm{sim}(C_A)$ using low-degree reconstruction. Recall $d \leq \Delta(C_A) \leq k^4 + k^3$. Learn $C_A|_\sigma$.

Learning Cluster

- Cluster is isolated after at most k settings $(\sigma := (X_{j_1} = \bar{\alpha}_{j_1}), \ldots)$
- Factor and Reconstruct $\mathrm{sim}(C_A)$ using low-degree reconstruction. Recall $d \leq \Delta(C_A) \leq k^4 + k^3$. Learn $C_A|_\sigma$.
- Learn

$$C_A = \frac{\prod_{X_j \in \sigma} \ell_{1,j}(X_j)}{\prod_{X_j \in \sigma} \ell_{1,j}(\bar{\alpha}_j)} \cdot C_A|_\sigma$$

UNIVERSITY OF TORONTO

Learning Full Circuit

- Learn $C' = C - C_A$.
- Check with PIT if the output circuit computes the same polynomial as C.

Questions?

Thank You.

M. Ben-Or and P. Tiwari.
A deterministic algorithm for sparse multivariate polynominal interpolation.
In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 301–309, 1988.

Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich.
Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits.
In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 809–822, 2021.

Zeyu Guo and Rohit Gurjar.
Improved explicit hitting-sets for roabps.
In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.

A. Gupta, P. Kamath, N. Kayal, and R. Saptharishi.
Arithmetic circuits: A chasm at depth three.

In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 578–587, 2013.

D. Yu. Grigor'ev and N.N. Vorobjov.
Solving systems of polynomial inequalities in subexponential time.
Journal of Symbolic Computation, 5(1):37 – 64, 1988.

Johan Håstad.
Tensor rank is np-complete.
J. Algorithms, 11(4):644–654, 1990.

D. Ierardi.
Quantifier elimination in the theory of an algebraically-closed field.
In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 138–147, New York, NY, USA, 1989. Association for Computing Machinery.

E. Kaltofen.
Single-factor hensel lifting and its application to the straight-line complexity of certain polynomials.

In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 443–452, 1987.

A. Klivans and D. Spielman.
Randomness efficient identity testing of multivariate polynomials.
In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

Z. S. Karnin and A. Shpilka.
Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in.
In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 274–285, 2009.

A. R. Klivans and A. A. Sherstov.
Cryptographic hardness for learning intersections of halfspaces.
J. Comput. Syst. Sci., 75(1):2–12, 2009.

J. Landsberg.
Tensors: geometry and applications.
Representation theory, 381(402):3, 2012.

Shir Peleg, Amir Shpilka, and Ben Lee Volk.
Tensor Reconstruction Beyond Constant Rank.
In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287, pages 87:1–87:20, 2024.

A. Shpilka.
Interpolation of depth-3 arithmetic circuits with two multiplication gates.
In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 284–293, 2007.

M. Schaefer and D. Stefankovic.
The complexity of tensor rank.
CoRR, abs/1612.04338, 2016.

A. Shpilka and I. Volkovich.
On the relation between polynomial identity testing and finding variable disjoint factors.
In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010.

UNIVERSITY OF TORONTO

Full version at https://eccc.weizmann.ac.il/report/2010/036.