Algorithmic and theoretical aspects of sparse deep neural networks

Quoc Tung Le

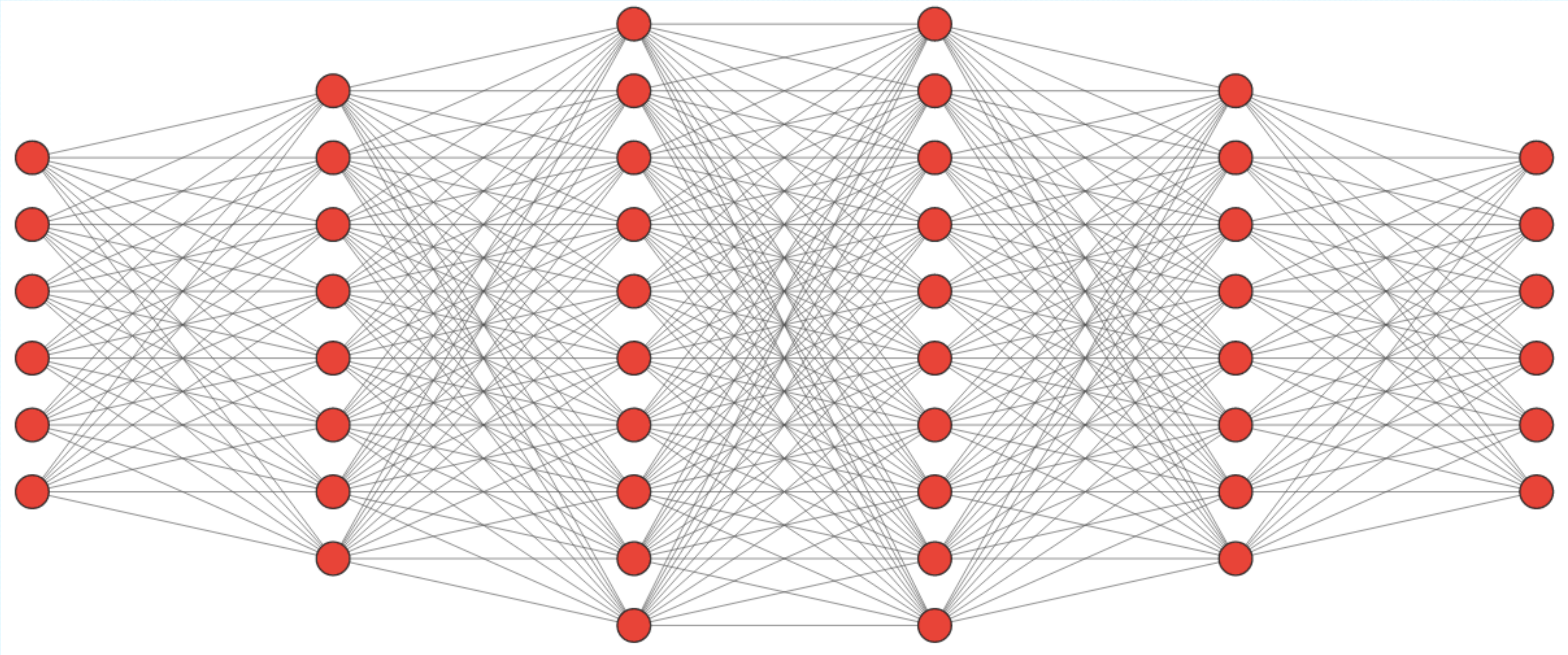
July 26th, 2023

Léon Zheng

Elisa Riccietti

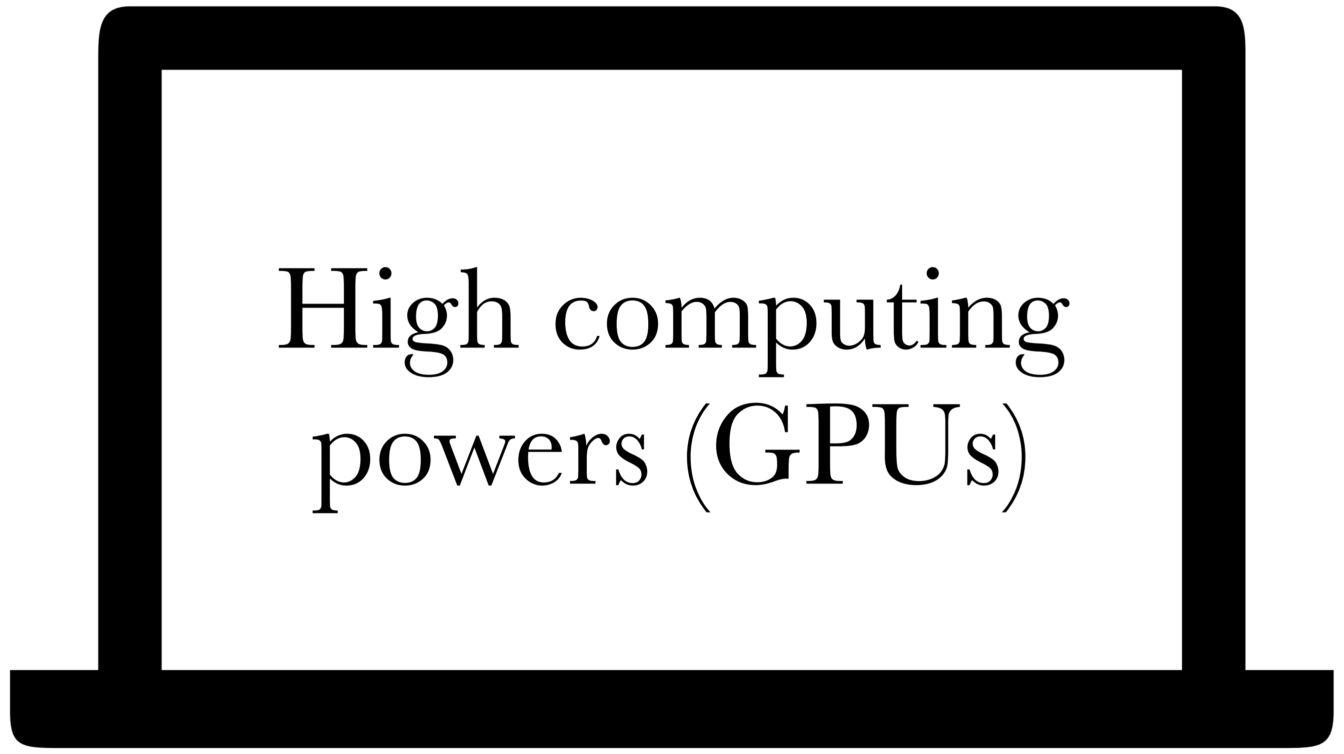
Rémi Gribonval

Deep learning and the recipe of success



Huge parametric models
(over-parameterization)

Massive data set



Deep learning and the recipe of success

Huge parametric models

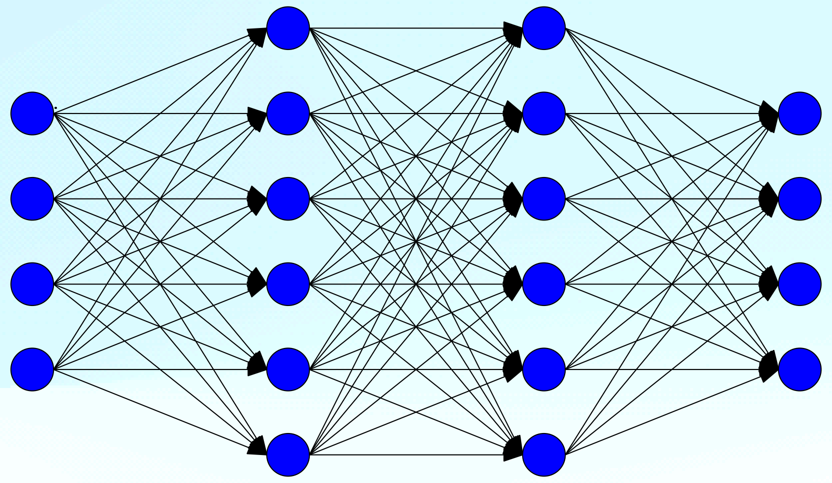
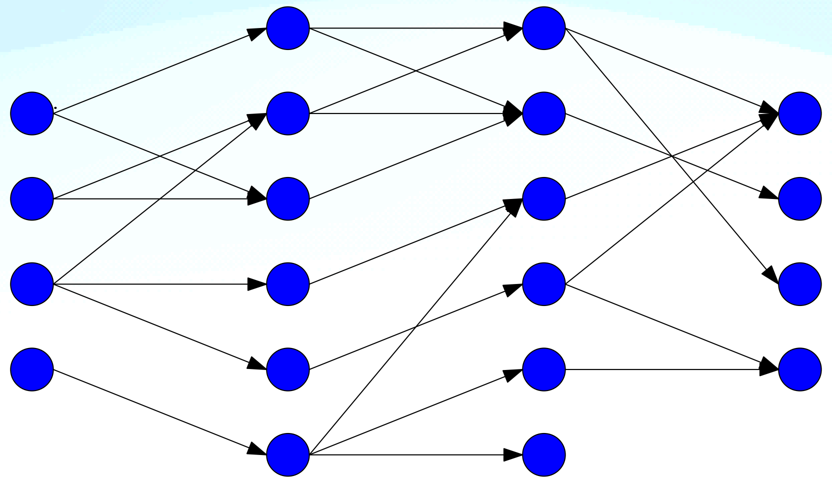
Massive dataset

High computing powers (GPUs)

❓ Can we make these three components **scalable**?

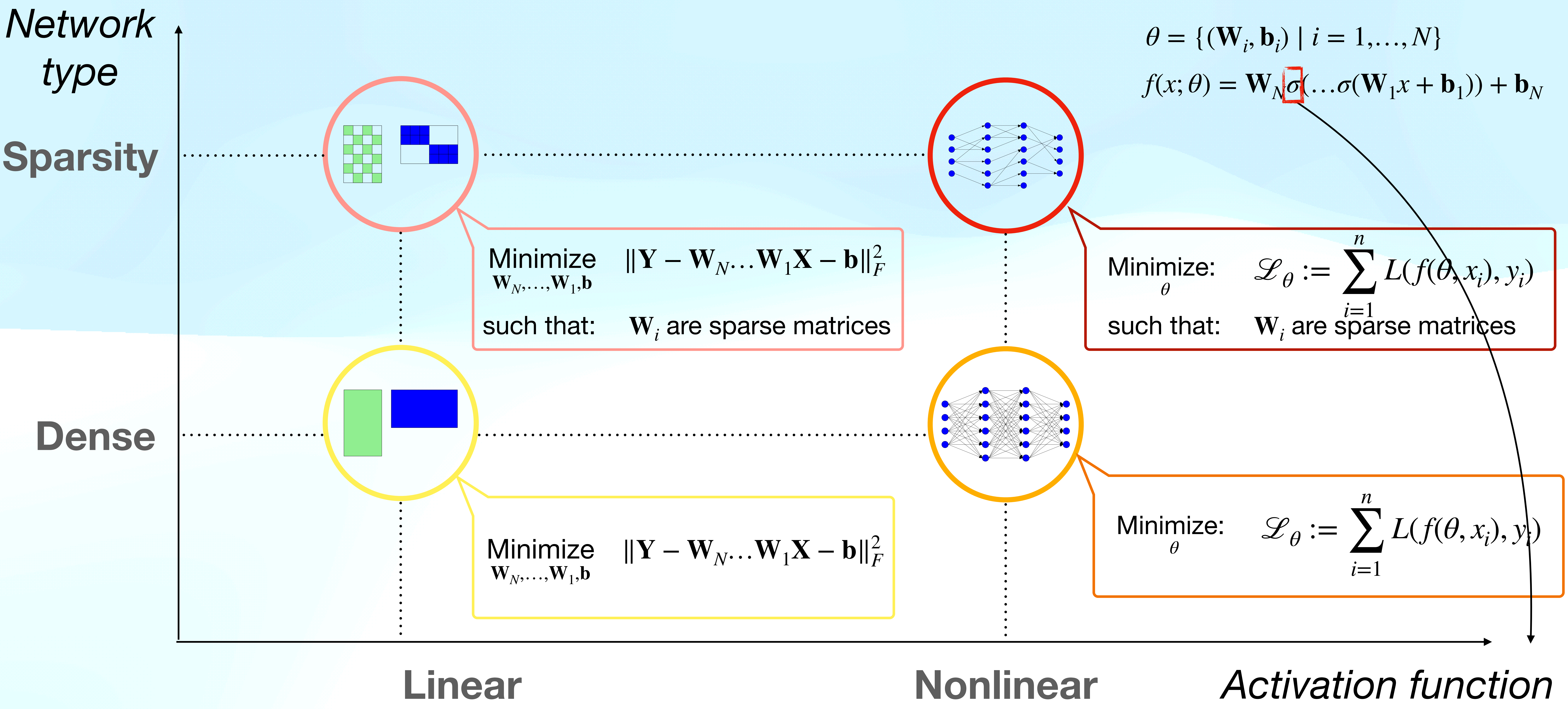
Sparsity: a good old friend

Sparse Deep Neural Networks

Usual Deep Neural Networks	Sparse Deep Neural Networks
	
× Over-parameterization → storage problems × Extremely data hungry → large training sets × Memorization of training data → privacy leakage	☑ Small parametric models (with high sparsity) ☑ Fewer learned parameters → fewer training data ☑ Reduce privacy issue

Sounds good. Then how can I train a sparse neural network?

Sparsity in Deep Neural Networks



Questions for Sparse Neural Networks

Existence of optimal solutions

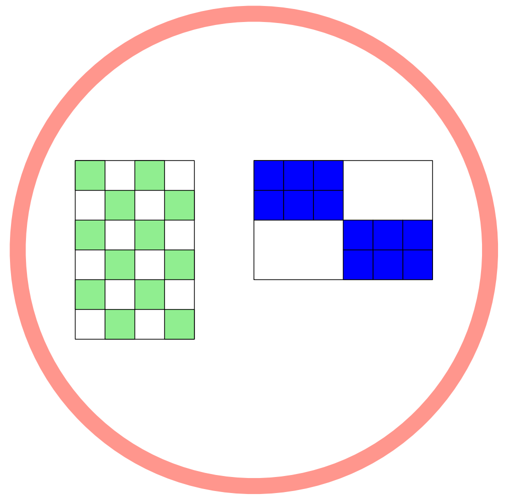
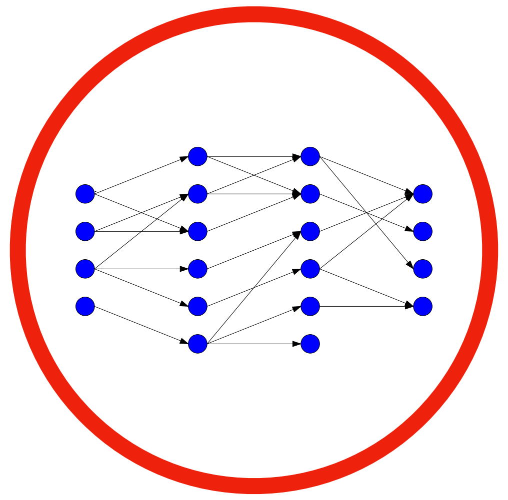
❓ Does the training problem of sparse neural networks **always admit an optimal solution**?

Tractability

❓ Is it **polynomially tractable** to train sparse neural networks?

Landscape

❓ What does the **landscape** of loss function look like? (e.g., does it have **local minima**?, etc.)

Strategy: **First**  **Then** 

From sparsity in Deep Learning to Matrix Factorisation

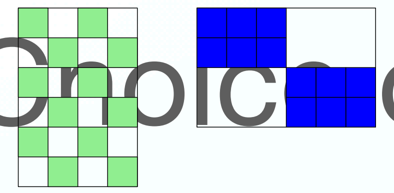
Sparse Matrix Factorisation

Given A and $\mathcal{E}_j$ some sets of ***sparse*** matrices, solve:

$$\min_{W^{(N)},\ldots,W^{(1)}} \|A - \prod_{j=1}^{N} W^{(j)}\|_F^2 \text{ subject to: } W^{(j)} \in \mathcal{E}_j, \forall j \in \{1,\ldots,N\}$$

Choice of sparse matrices set $\mathcal{E}_j$

- k-sparse per row,
- k-sparse per column
- k-sparse in total



= Linear Sparse Neural Network ≃ Sparse Matrix Factorisation

Linear Sparse Neural Network	Sparse Matrix Factorisation
$\text{Minimize}_{\mathbf{W}_N,\ldots,\mathbf{W}_1,\mathbf{b}} \quad \|\mathbf{Y} - \mathbf{W}_N\ldots\mathbf{W}_1\mathbf{X} - \mathbf{b}\|_F^2$	$\text{Minimize}_{\mathbf{W}_N,\ldots,\mathbf{W}_1} \quad \|\mathbf{A} - \mathbf{W}_N\ldots\mathbf{W}_1\|_F^2$

Sparse matrix factorization

Fixed support matrix factorization

Special case of sparse matrix factorization

SPARSE MATRIX FACTORISATION

$$\min_{W^{(N)},\ldots,W^{(1)}} \|A - \prod_{j=1}^{N} W^{(j)}\|_F^2 \text{ subject to: } W^{(j)} \in \mathscr{E}_j, \forall j \in \{1,\ldots,N\}$$

- $N = 2$
- $(\mathscr{E}_1, \mathscr{E}_2)$: set of matrices whose **supports** are included in **given sets** I and J

FIXED SUPPORT MATRIX FACTORISATION

$$\min_{X,Y} \|A - XY^\top\|_F^2 \text{ subject to: } \mathrm{supp}(X) \subseteq I, \mathrm{supp}(Y) \subseteq J$$

Fixed support matrix factorization (FSMF)

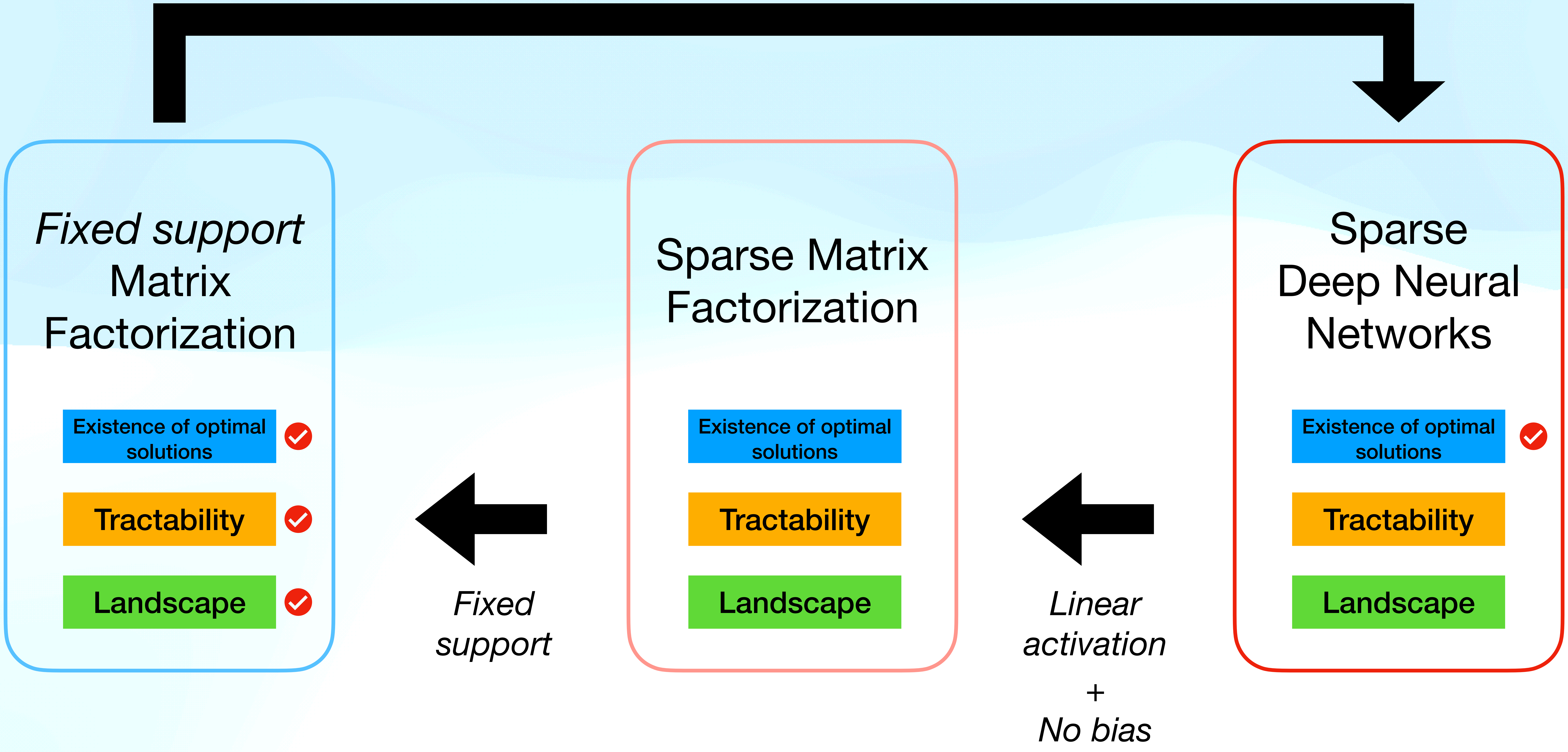
$$\min_{X,Y} \|A - XY^\top\|_F^2 \text{ subject to: } \operatorname{supp}(X) \subseteq \boxed{I}, \operatorname{supp}(Y) \subseteq \boxed{J}$$

SUPPORT CONTRAINTS

$$\mathbf{A} \approx \overset{\mathbf{X}}{\begin{bmatrix} ? & ? & 0 & ? \\ 0 & ? & ? & 0 \\ ? & 0 & ? & 0 \\ 0 & ? & 0 & ? \end{bmatrix}} \times \overset{\mathbf{Y}^\top}{\begin{bmatrix} 0 & ? & 0 & ? & 0 \\ ? & ? & 0 & 0 & ? \\ ? & 0 & ? & ? & 0 \\ ? & 0 & ? & 0 & ? \end{bmatrix}}$$

inside support   outside support

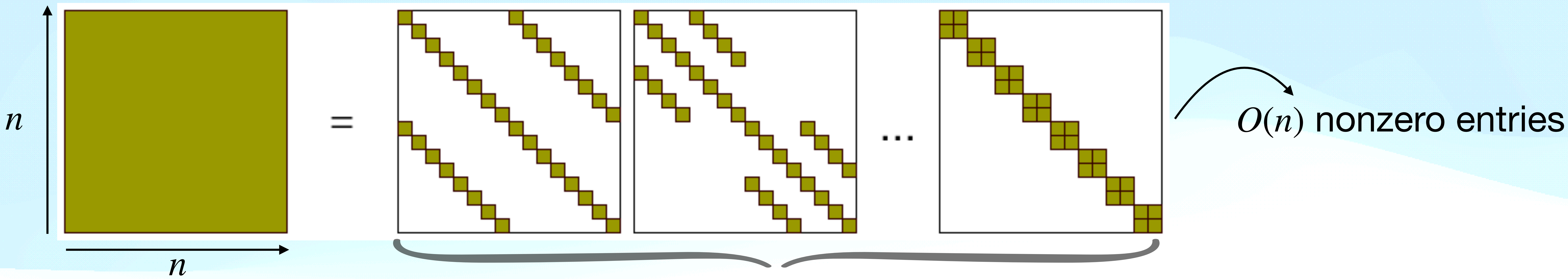
Overview of our reasoning



Further motivation for sparse matrix factorization

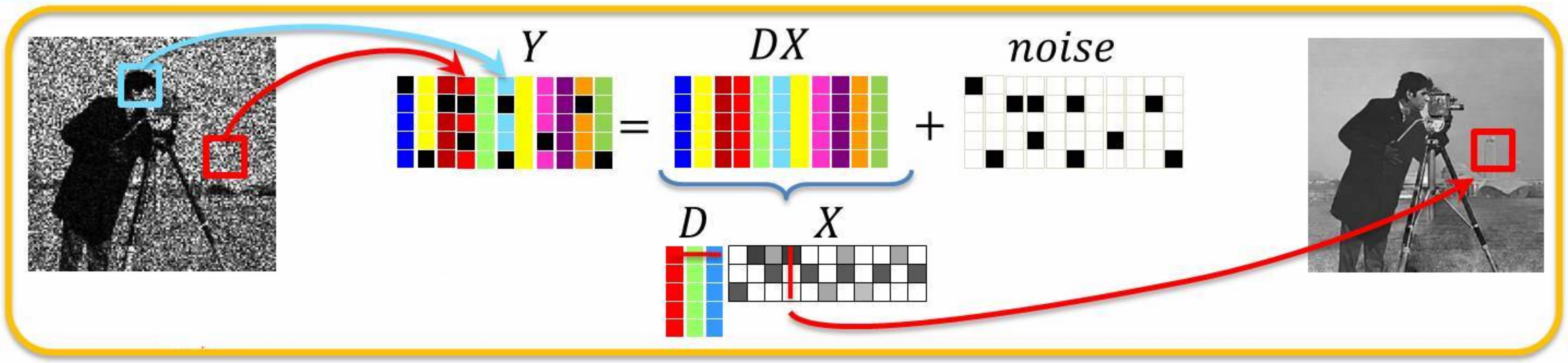
Why sparse matrix factorisation?

Fast linear operator: if $\mathbf{A} \approx \mathbf{W}_1 \ldots \mathbf{W}_J$ then $\mathbf{Ax} = \mathbf{W}_1 \ldots \mathbf{W}_J \mathbf{x}$



The Discrete Fourier Transformation sparse factorisation and its $O(n \log n)$ fast algorithm

Dictionary learning: given a dataset $\mathbf{Y}$, find atoms $\mathbf{D}$ and look-up table $\mathbf{X}$



Plan of the talk

Fixed Support Matrix Factorisation

Existence of optimal solutions in sparse ReLU neural networks training

Butterfly parameterization in sparse deep neural networks

Fixed support matrix factorization

Results on (FSMF)

$$\min_{X,Y} \quad L(X,Y) = \|A - XY^\top\|_F^2 \ \text{ subject to: } \ \mathrm{supp}(X) \subseteq I, \mathrm{supp}(Y) \subseteq J$$

Existence of optimal solutions

☞ Does (FSMF) **always admit an optimal solution**?

Tractability

☞ Is (FSMF) **polynomially tractable**?

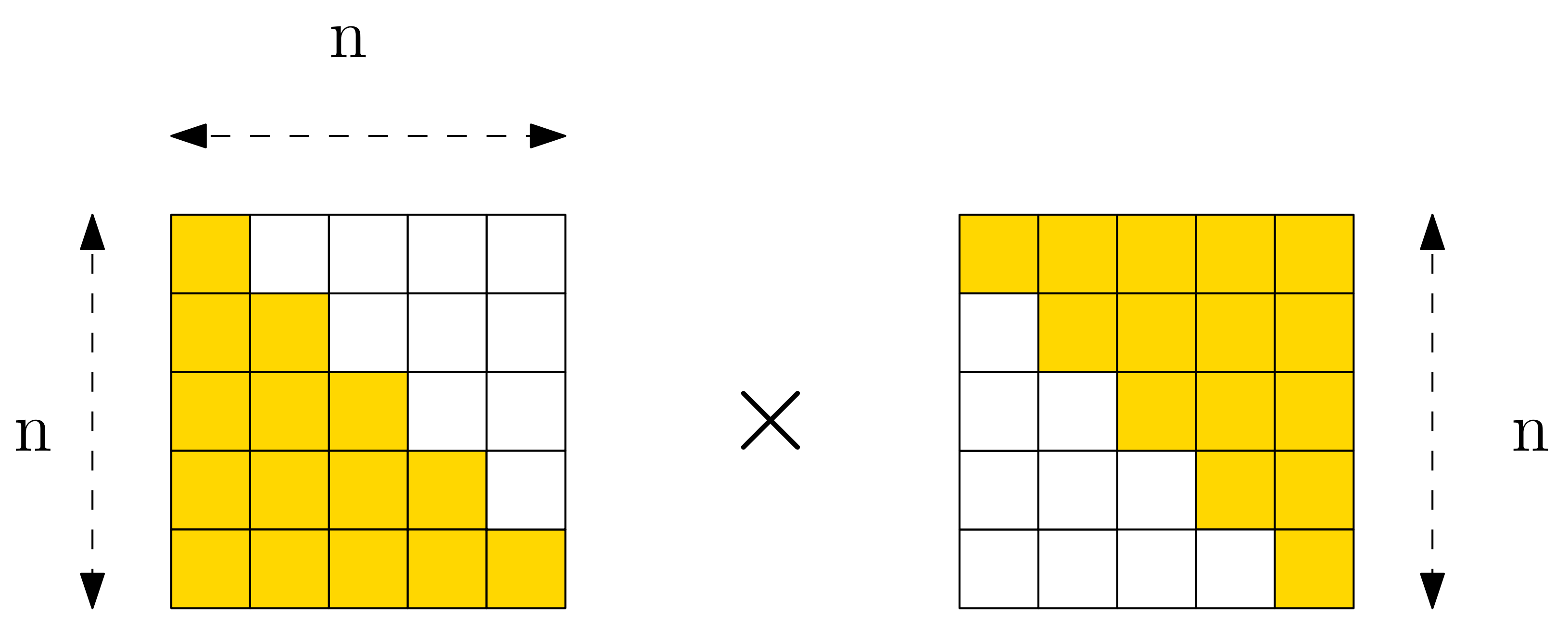
Landscape

☞ What does the **landscape** of $L(X,Y)$ look like?
(e.g., does it have **local minima**?, etc.)

Non-existence of optimal solutions

(FSMF) does not always admit an optimal solution

$A =$



$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$n = 2$$

☞ Infimum is **zero**: $X_k = \begin{pmatrix} \frac{1}{k} & 0 \\ 1 & -1 \end{pmatrix}, Y_k^\top = \begin{pmatrix} 1 & k \\ 0 & k \end{pmatrix}, \lim X_k Y_k^\top = A\,.$

☞ Infimum is **not** attained: There is no feasible (X, Y) such that $A = XY^\top$.

(Gene H. Golub and Charles F. Van Loan, Matrix Computations)

Equivalence between existence - closedness

ORIGINAL FORMULATION

$$\min_{X,Y} \|A - XY^\top\|_F^2 \text{ subject to: } \mathrm{supp}(X) \subseteq I, \mathrm{supp}(Y) \subseteq J$$

Change of variables

EQUIVALENT FORMULATION

$$\min_{B \in \mathscr{P}_{I,J}} \|A - B\|_F^2 \text{ where } \mathscr{P}_{I,J} := \{XY^\top \mid \mathrm{supp}(X) \subseteq I, \mathrm{supp}(Y) \subseteq J\}$$

PROJECTION OF A ONTO THE SET $\mathscr{P}_{I,J}$

Optimal solutions exist if and only if $\mathscr{P}_{I,J}$ is closed

Deciding the closedness of $\mathscr{P}_{I,J}$

Given (I, J), decide the **closedness** of $\mathscr{P}_{I,J}$.

REMINDER: $\mathscr{P}_{I,J} := \{XY^\top \mid \text{supp}(X) \subseteq I, \text{supp}(Y) \subseteq J\}$

$\mathscr{P}_{I,J}$ is closed? $\Longleftrightarrow$ $\overline{\mathscr{P}_{I,J}} \setminus \mathscr{P}_{I,J}$ is empty?

closure of $\mathscr{P}_{I,J}$

semi-algebraic set

👉 The emptyness of a semi-algebraic set is decidable (using Quantifier Elimination Algorithm). (S. Basu, R. Pollack, M-F Roy, Algorithms in Real Algebraic Geometry)

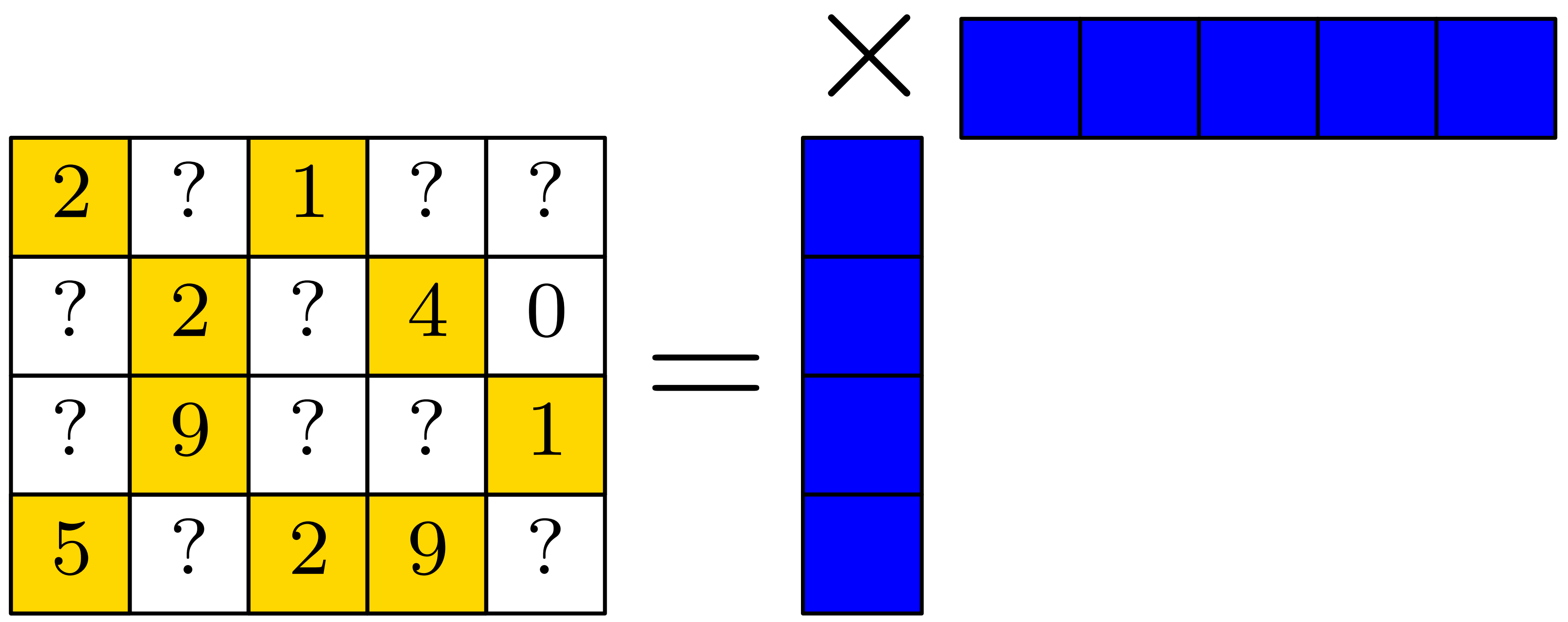
👉 The complexity is doubly exponential (w.r.t. the sizes of I, J and the matrix).

NP-hardness

THEOREM I

For *arbitrary support constraint* (I, J), (FSMF) is NP-hard.

PROOF: $\boxed{\text{Rank-one matrix completion}}$ is reducible to (FSMF).



This problem is NP-hard.

(N.Gillis, F. Glineur, SIAM Journal on Matrix Analysis and Applications)

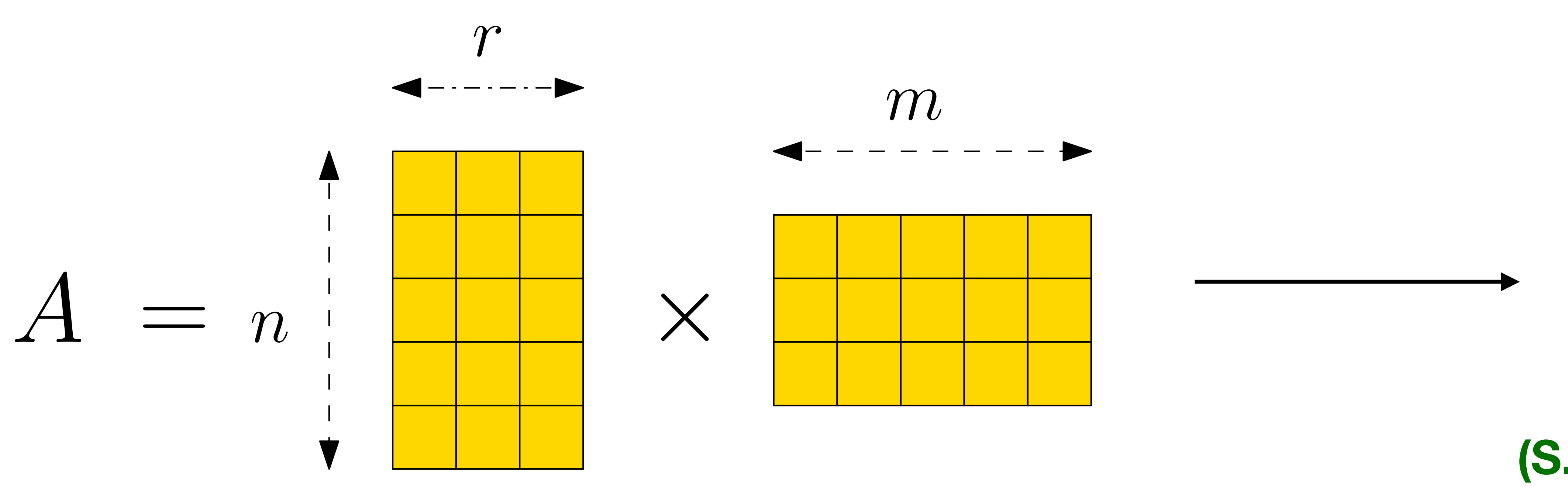
Tractability with structured supports

An example of tractable instances

Unconstrained Matrix Factorisation

When there is **no** constraint on the supports of (X, Y)

$$\underset{X \in \mathbb{R}^{n \times r}, Y \in \mathbb{R}^{m \times r}}{\text{Minimize}} \; L(X, Y) = \|A - XY^\top\|_F^2$$



Best rank r approximation of the matrix A.

(S. Burer, R. D.C. Monteiro, Mathematical Programming)

Algorithm: Using (Truncated) Singular Value Decomposition.

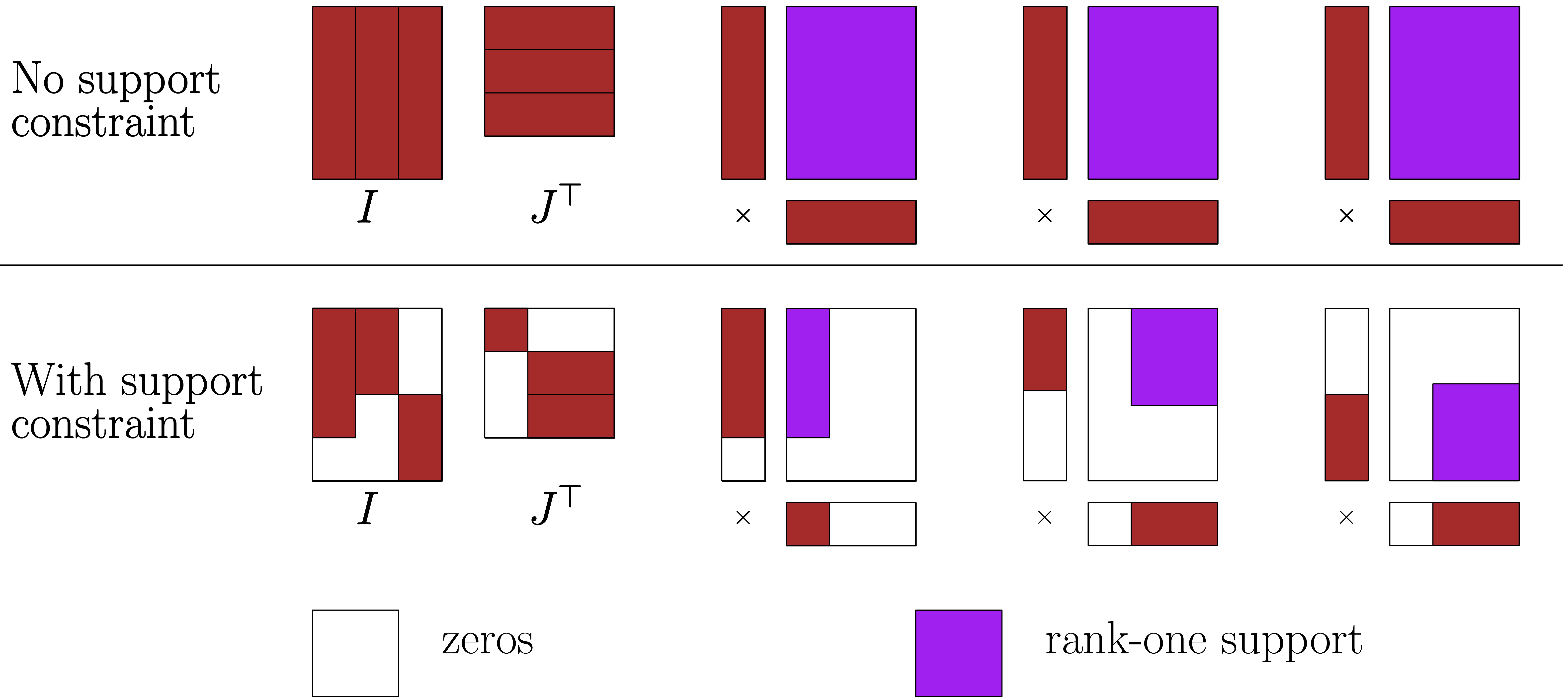
Can (Truncated) Singular Value Decomposition still work in constrained cases?

Tractability with structured supports (cont)

Rank one contribution supports

kth column of X rank at most 1 kth column of Y

$$XY^\top = \boxed{X_1 Y_1^\top} + \boxed{X_2 Y_2^\top} + \boxed{X_3 Y_3^\top}$$

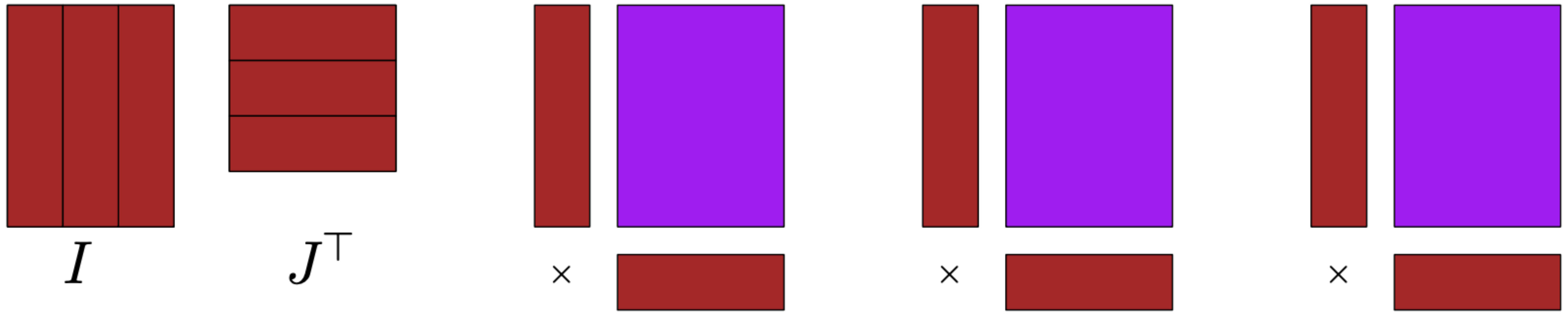


Tractability with structured supports (cont)

What is special about unconstrained matrix factorization?

$$XY^\top = X_1Y_1^\top + X_2Y_2^\top + X_3Y_3^\top$$

No support constraint



!! All rank-one supports are **identical**

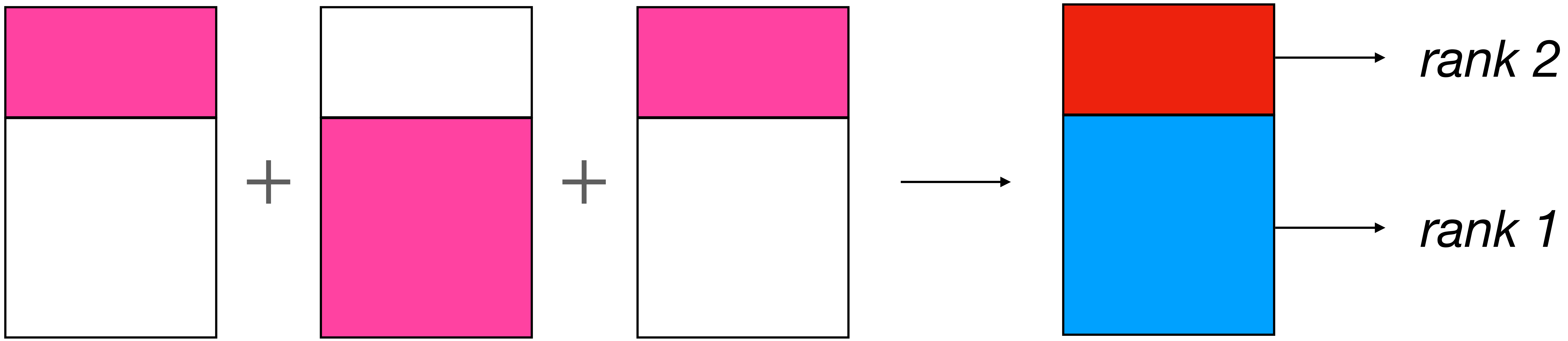
THEOREM II

If all rank-one supports are **pairwise disjoint** or **identical**, then the corresponding instance of (FSMF) is polynomially tractable.

Tractability with structured supports (cont)

THEOREM II

If all rank-one supports are **pairwise disjoint** or **identical**, then the corresponding instance of (FSMF) is polynomially tractable.



rank-one supports

☞ Algorithm: Using (Truncated) Singular Value Decomposition for **submatrices** of the target matrix.

Litteratures on the landscape of $L(X, Y)$

$$L(X, Y) = \|A - XY^\top\|_F^2$$

Has been studied for:

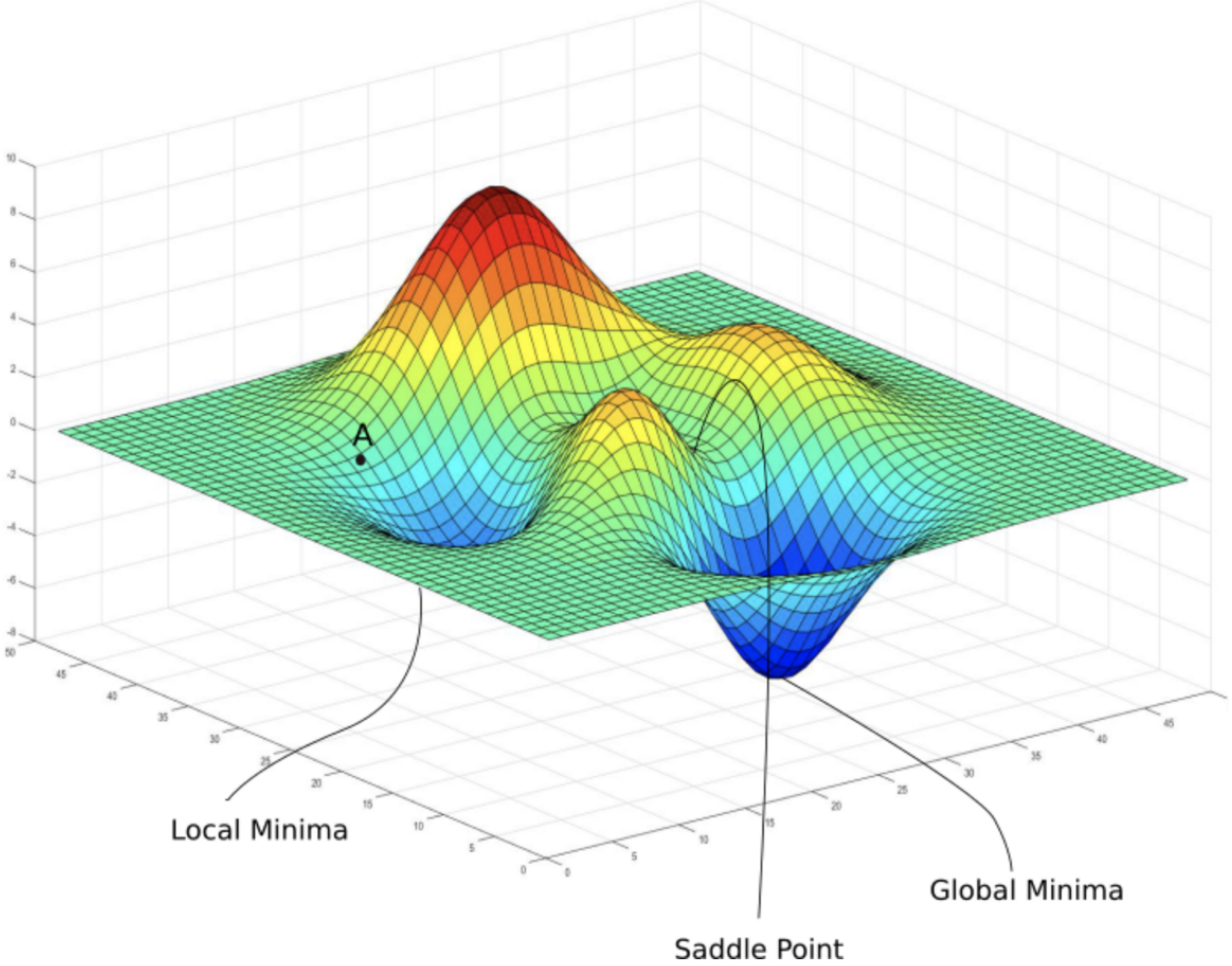
👉 Linear and shallow neural networks

👉 Matrix sensing, phase retrieval, matrix completion.

(Q. Li, Z. Zhu, G. Tang, The nonconvex geometry of low-rank matrix optimization, Information and Inference, 2018)

(Z. Zhu, D. Soudry, Y.C. Eldar, M.B. Wakin, The global optimization geometry of shallow linear neural networks, JMIV, 2019)

(L. Venturi, A. S. Bandeira, J. Bruna, Spurious valleys in one-hidden-layer neural network optimization landscapes, JMLR, 2019)



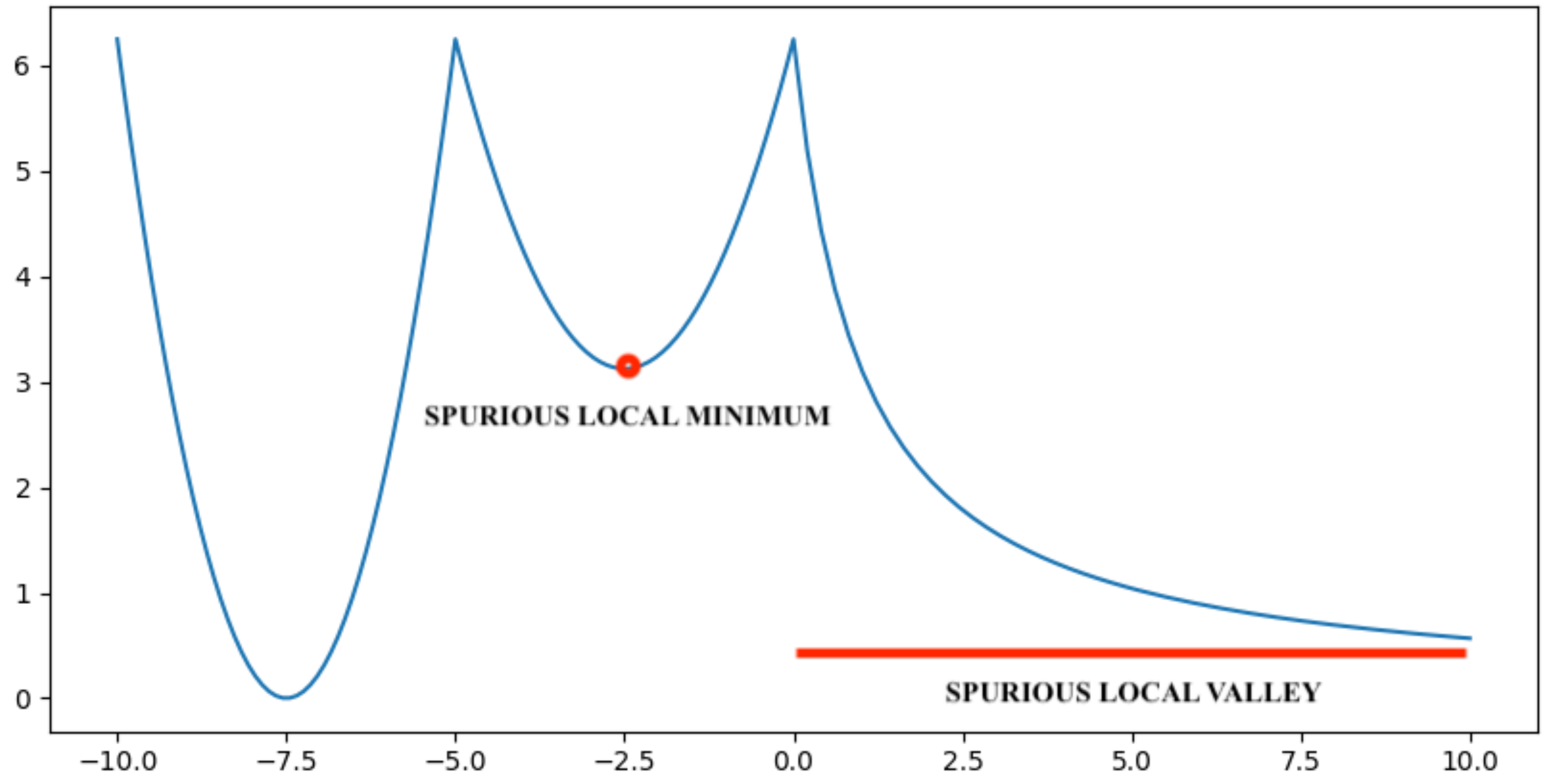
❓ What does the landscape look like in the constrained cases?

Unconstrained matrix factorization

$$L(X, Y) = \|A - XY^\top\|_F^2$$

☞ There is no spurious local minimum for any A.

☞ There is no spurious local valley for any A.



? Do these properties still hold in constrained cases?

Benign landscape of tractable instances

Reminder: Fixed Support Matrix Factorization

$$\min_{X,Y} \|A - XY^\top\|_F^2 \quad \text{subject to:} \quad \text{supp}(X) \subseteq I, \text{supp}(Y) \subseteq J$$

THEOREM III

If (I, J) satisfies the condition of **Theorem II**, then there is no spurious local minima and spurious local valleys.

Summary on (FSMF)

Existence of optimal solutions

- There are instances (A, I, J) which (FSMF) admits no optimal solution.

NP-hardness

- For arbitrary (I, J), (FSMF) is *NP-hard* to solve.

Tractability

- For certain structured (I, J), (FSMF) has a polynomial algorithm.

Benign landscape

- With the same family of structured (I, J), loss function of (FSMF) has no local minima.

(Q-T. Le, E. Riccietti, R. Gribonval, SIAM Journal of Matrix Analysis and Applications, 2023)

Existence of optimal solutions in ReLU sparse neural network training

Sparse ReLU Neural Networks Training

Optimization problem of Sparse Neural Networks

Given data set $\mathscr{D} := (X, Y)$ and $\mathscr{E}_j$ some sets of ***sparse*** matrices, solve:

$$\underset{W^{(j)}, b^{(j)}}{\text{Minimize}} \quad \|Y - W^{(N)}\sigma(\ldots\sigma(W^{(1)}X + b^{(1)}) + \ldots) + b^{(N)}\|_F^2$$

$$\text{subject to:} \quad W^{(j)} \in \mathscr{E}_j, \forall j \in \{1,\ldots,N\}$$

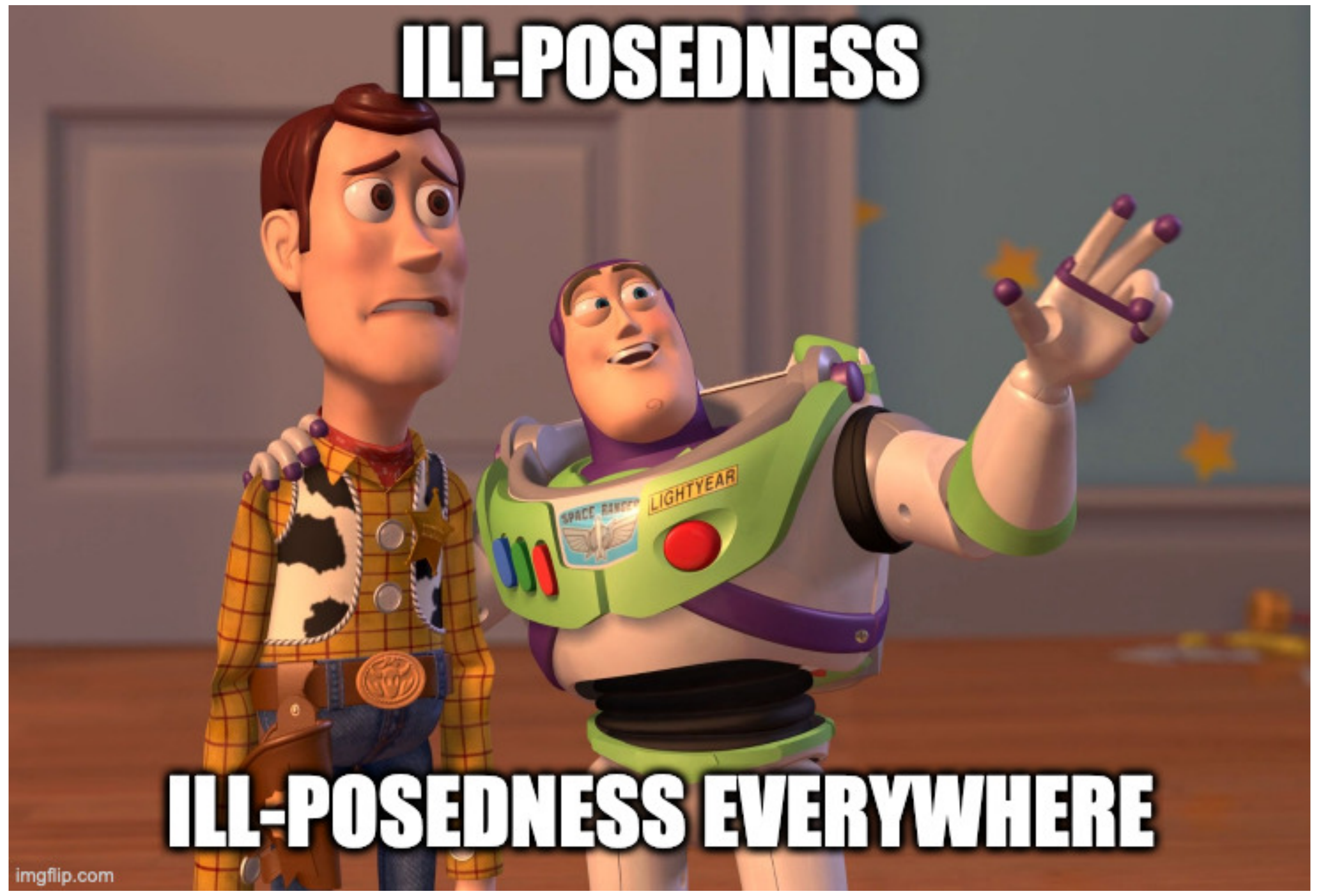
☞ σ is the **ReLU** activation function: $\sigma(x) = \max(x,0)$.

☞ In practice, $\mathscr{E}_j$ is usually chosen as the set of $\mathbf{k}$**-sparse matrices**.

(J. Frankle, M. Carbin, ICLR 2019), (S. Han, H. Mao, W-J. Dally, ICLR 2016)

☞ We consider **quadratic loss function** for simplification. Our argument works for any **coercive** loss function.

Non-existence of optimal solutions - ill-posedness



Tensor decomposition (order at least three)	TENSOR RANK AND THE ILL-POSEDNESS OF THE BEST LOW-RANK APPROXIMATION PROBLEM VIN DE SILVA* AND LEK-HENG LIM†
Matrix Completion	Low-Rank Matrix Approximation with Weights or Missing Data is NP-hard Nicolas Gillis[1] and François Glineur[1]
Robust Principle Component Analysis	Matrix rigidity and the ill-posedness of Robust PCA and matrix completion* Jared Tanner†‡ Andrew Thompson§ Simon Vary†
(Classical) Neural Network Training	**Best k-Layer Neural Network Approximations** Lek-Heng Lim[1] · Mateusz Michałek[2,3] · Yang Qi[4]

How about the training problem of sparse ReLU neural networks?

Fixed support sparse ReLU neural networks

Given data set $\mathscr{D} := (X, Y)$, solve:

GENERAL

$$\min_{W^{(j)}, b^{(j)}} \quad \|Y - W^{(L)}\sigma(\ldots\sigma(W^{(1)}X + b^{(1)}) + \ldots) + b^{(L)}\|_F^2$$

subject to: $\boxed{W^{(j)} \in \mathscr{E}_j,} \forall j \in \{1,\ldots,N\}$

FIXED SUPPORT

$$\min_{W^{(j)}, b^{(j)}} \quad \|Y - W^{(N)}\sigma(\ldots\sigma(W^{(1)}X + b^{(1)}) + \ldots) + b^{(N)}\|_F^2$$

subject to: $\boxed{\text{supp}(W^{(j)}) \in I_j,} \forall j \in \{1,\ldots,N\}$

DÉJÀ VU: closedness vs existence of optimal solutions

Given a support constraint $(I_1, \ldots, I_N)$, **does** optimal solutions always exist **for all data set** $\mathscr{D}$ for the corresponding training problem ?

The support constraint $(I_1, \ldots, I_N)$ makes the training problem **always admit optimal solutions** if and only if **for all** input sets X, the image of the function $\theta := \{(W^i, b^i)\} \mapsto W^{(N)}\sigma(\ldots\sigma(W^{(1)}X + b^{(1)}) + \ldots) + b^{(N)}$ is **closed**.

Sufficient condition for the existence of optimal solutions

THEOREM IV

For **two-layer** neural networks ($N = 2$) with **output dimension** equal to **one**, any support constraint makes the training problem **always admit optimal solutions**.

(Q-T. Le, E. Riccietti, R. Gribonval, preprint, 2023)

COROLLARY I

For **two-layer** neural networks ($N = 2$) with **output dimension** equal to **one**, the constraints $\mathcal{E}_j := \{X \mid \|X\|_0 \leq k_j\}, j = 1{,}2$ makes the training problem **always admit optimal solutions**.

Necessary condition for the existence of optimal solutions

THEOREM V

For **two-layer** neural networks ($N = 2$) with support constraint (I, J), if the training problem always admits optimal solutions, then $\mathscr{P}_{I,J}$ is **closed**.

(Q-T. Le, E. Riccietti, R. Gribonval, preprint, 2023)

this is decidable

$$\mathscr{P}_{I,J} := \{XY^\top \mid \text{supp}(X) \subseteq I, \text{supp}(Y) \subseteq J\}$$

THEOREM VI

For fixed support neural networks with support constraint $(I_1, \ldots, I_N)$, if the training problem always admits optimal solutions, then $\mathscr{P}_{I_1,\ldots,I_N}$ is **closed**.

Necessary condition for the existence of optimal solutions

THEOREM

For **two-layer** neural networks ($N = 2$) with support constraint (I, J), if the training problem always admits optimal solutions, then $\mathscr{P}_{I,J}$ is **closed**.

The condition is just necessary because when there is **no constraint** on the support, the training problem is ill-posed for certain data set.

(L-H. Lim, M. Michalek, Y. Qi, Constructive Approximation 2019)

Butterfly parameterization in sparse neural networks

Introduction to butterfly parameterization

Discrete Fourier Transform (DFT): $$y_k = \sum_{n=0}^{N-1} e^{-\frac{i2\pi}{N}kn} x_n, \qquad k = 0,\ldots,N-1$$

$$y = F_N x \qquad F_N = \left(e^{-\frac{i2\pi}{N}kn}\right)_{k,n\in\{0,\ldots,N-1\}}$$

Algebraic properties of the Fourier kernel:

$$y_k = \boxed{\sum_{m=0}^{N/2-1} e^{-\frac{i2\pi}{N/2}km} x_{2m}} + e^{-\frac{i2\pi}{N}k} \boxed{\sum_{m=0}^{N/2-1} e^{-\frac{i2\pi}{N/2}km} x_{2m+1}}$$

$$y_{k+N/2} = \boxed{\sum_{m=0}^{N/2-1} e^{-\frac{i2\pi}{N/2}km} x_{2m}} - e^{-\frac{i2\pi}{N}k} \boxed{\sum_{m=0}^{N/2-1} e^{-\frac{i2\pi}{N/2}km} x_{2m+1}}$$

The Cooley - Tukey algorithm (radix-2)

DFT on **even** indices

DFT on **odd** indices

Matrix factorization of DFT

$$F_N x = \begin{pmatrix} F_{N/2}x_e + D_{N/2}F_{N/2}x_o \\ F_{N/2}x_e - D_{N/2}F_{N/2}x_o \end{pmatrix}$$

$$= \begin{pmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{pmatrix} \begin{pmatrix} F_{N/2} & \mathbf{0} \\ \mathbf{0} & F_{N/2} \end{pmatrix} \boxed{P_N} x$$

Permutation matrix

$$D_{N/2} = \begin{pmatrix} 1 & & & & \\ & e^{-\frac{i2\pi}{N}} & & & \\ & & e^{-\frac{i2\pi}{N}2} & & \\ & & & \ddots & \\ & & & & e^{-\frac{i2\pi}{N}\left(\frac{N}{2}-1\right)} \end{pmatrix}$$

Unrolling the recursion:

$$F_N = B_N \begin{pmatrix} F_{N/2} & \mathbf{0} \\ \mathbf{0} & F_{N/2} \end{pmatrix} P_N$$

$$= B_N \begin{pmatrix} B_{N/2} & \mathbf{0} \\ \mathbf{0} & B_{N/2} \end{pmatrix} \begin{pmatrix} B_{N/4} & & & \\ & B_{N/4} & & \\ & & B_{N/4} & \\ & & & B_{N/4} \end{pmatrix} \begin{pmatrix} P_{N/2} & \mathbf{0} \\ \mathbf{0} & P_{N/2} \end{pmatrix} P_N$$

$$= \ldots$$

Matrix factorization of DFT

Assume that $N = 2^L$:

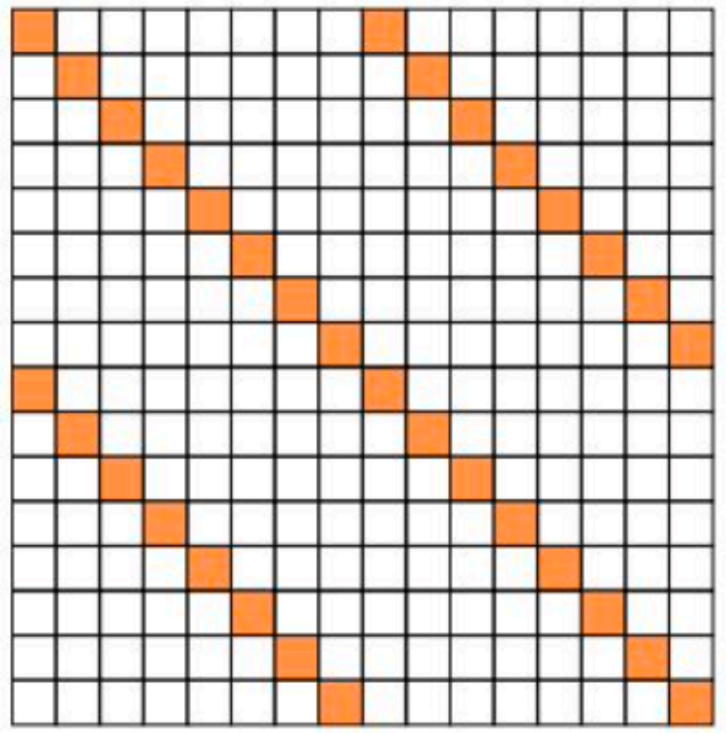
$$\boxed{F_N} = \boxed{X^{(1)} \ldots X^{(L)} P}$$

$$\text{supp}(X^{(\ell)}) \subseteq \boxed{\mathbf{I}_{2^{\ell-1}} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \otimes \mathbf{I}_{\frac{N}{2^{\ell}}}} \longrightarrow$$ *support constraint in binary matrix form*

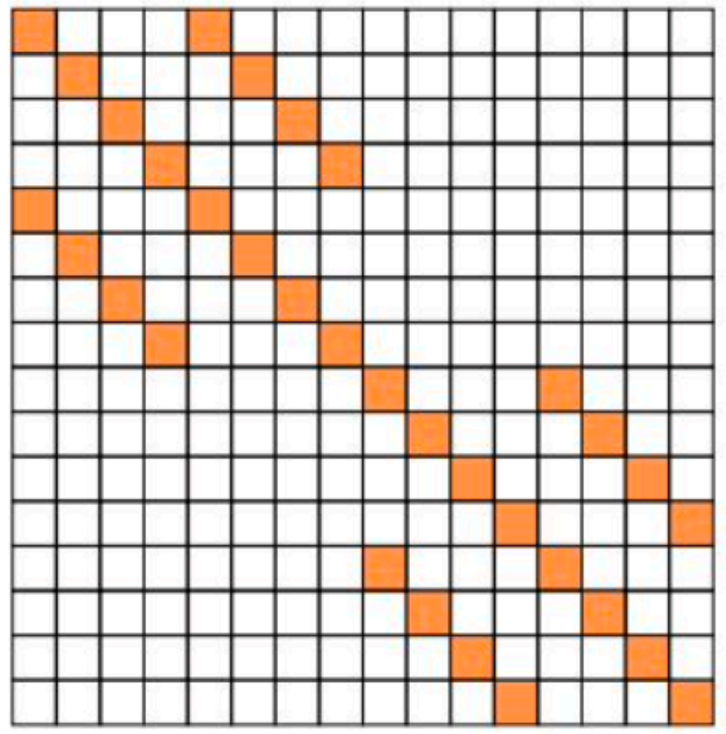
$O(N^2)$

$O(N \log N)$

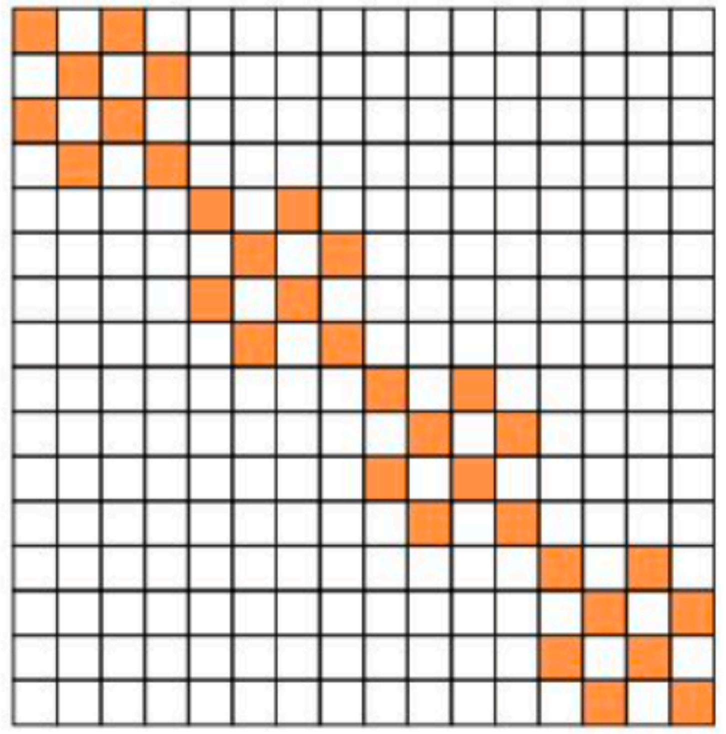
For $N = 16$:



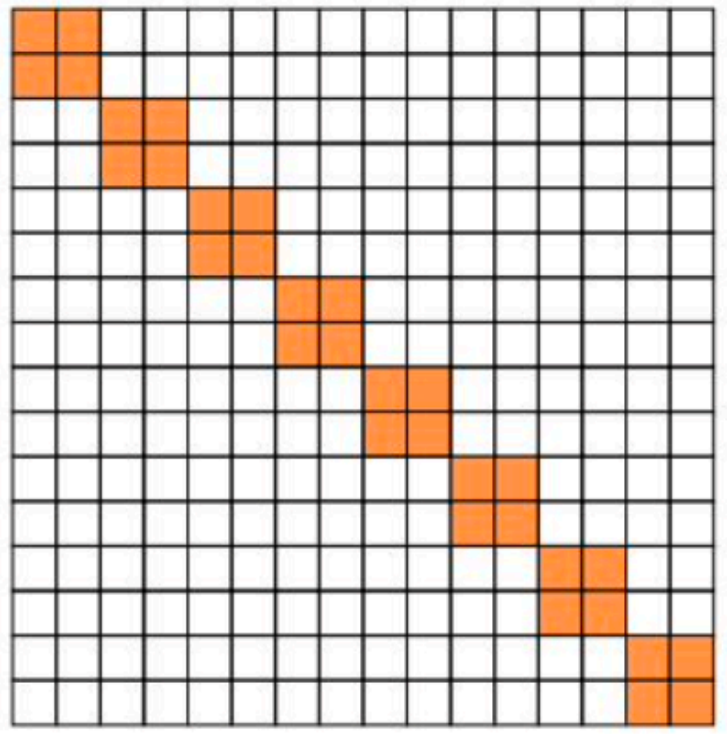
supp$(X^{(1)})$



supp$(X^{(2)})$



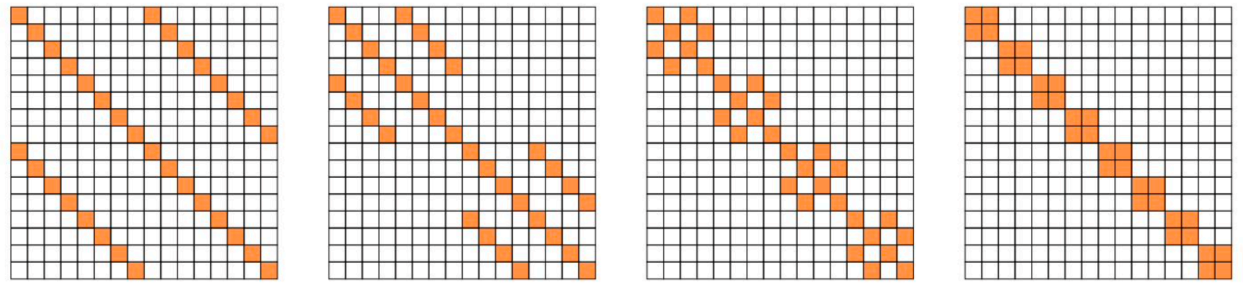
supp$(X^{(3)})$



supp$(X^{(4)})$

Butterfly parameterization in Sparse Neural Networks

Classical neural networks → *Butterfly* sparse neural networks

 $x + b$ →  $x + b$

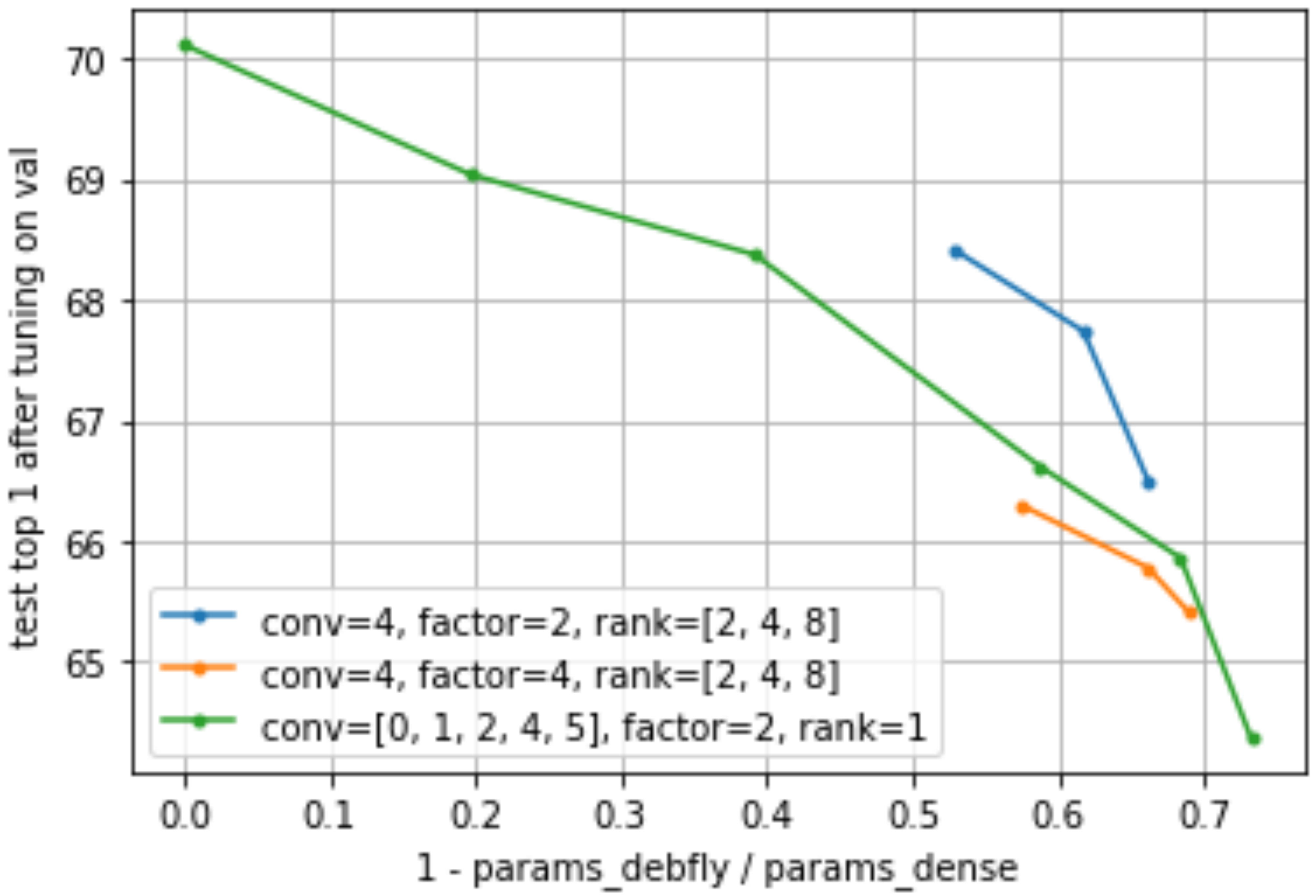
$O(N^2)$ → $O(N \log N)$

Parameterization	Number of factors	Matrix size	Introduced by
Butterfly	L	$2^L \times 2^L$	T. Dao et. al., 2019
Kaleidoscope	2L	$2^L \times 2^L$	T. Dao et. al., 2020
Monarch	2	$m \times n$	T. Dao et. al., 2022
Deformable butterfly	flexible	$m \times n$	R. Lin et. al., 2022

!! Supports of factors are **fixed**, **sparse** and very **structured**.

Interpretation of butterfly parameterization

❓ Among all existing parameterization, which one should we choose?



Trade-off between performance and compression

Approximation a matrix by butterfly parameterization

$$\min_{W^{(1)},\ldots,W^{(L)}} \|A - W^{(1)}\ldots W^{(L)}\|_F \qquad \text{s.t. } W^{(\ell)} \text{ is butterfly} \qquad (1)$$

- Generalized version of (FSMF) with *structured* supports.
- Existing algorithm: hierarchical factorization - **butterfly algorithm**.

(Michielssen & Boag, 1996); (O'Neil, Woolfe & Rokhlin, 2010); (Liu et. al. 2021)

⚠ No theoretical guarantee yet.

‼ **Hypothesis class of matrix:**

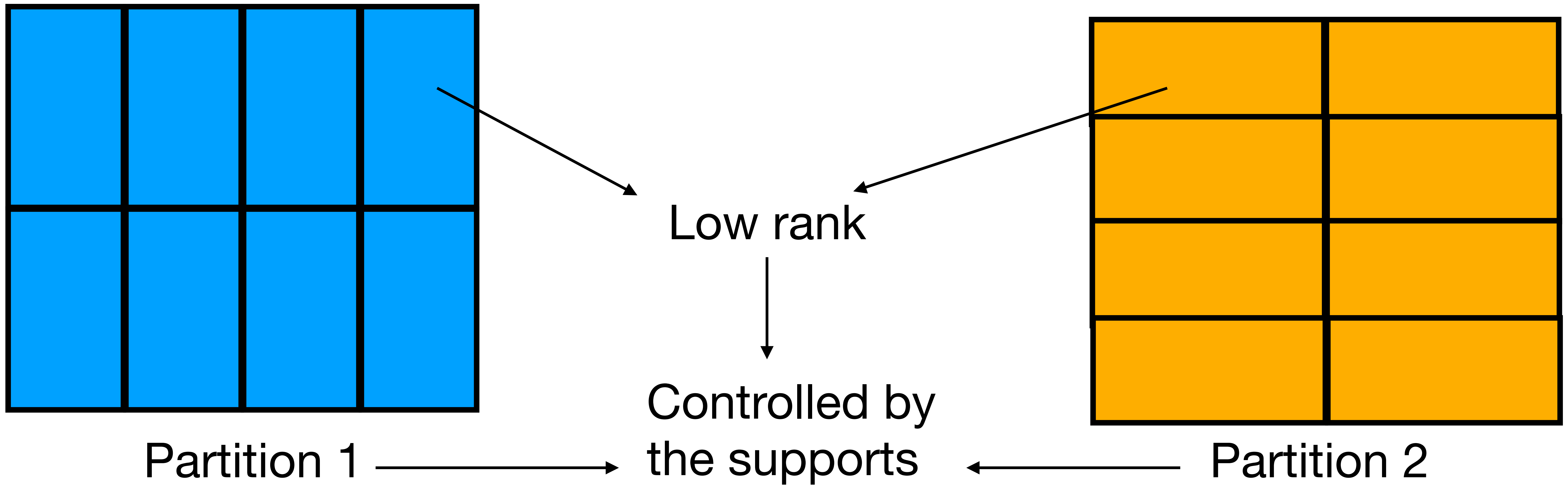
$$\mathscr{B} := \{W^{(1)}\ldots W^{(L)} \mid \text{Infimum of } (1) = 0\}$$

Analysis of butterfly parameterization

THEOREM VII

If E^* is the best error approximation of (1), the butterfly algorithm yields a solution whose distance to A is smaller than $(2^{L-1} - 1)E^*$

Algebraic description of $\mathscr{B}$:



☞ Also known as *complementary low-rank matrices* in the literature.

Contribution and future works

TAKE AWAY MESSAGE

- Link between sparse matrix factorization and its variant (FSMF) with sparse ReLU neural networks.
- Necessary/Sufficient condition for the existence of optimal solutions sparse ReLU neural networks.
- Butterfly parameterization in sparse deep neural networks

POSSIBLE IMPROVEMENT?

- Better algorithms to decide the ill-posedness of (FSMF).
- A full characterization of ill-posedness of sparse ReLU neural networks.

https://faust.inria.fr/

https://arxiv.org/abs/2112.00386

https://arxiv.org/abs/2306.02666

THANK YOU

Analysis of butterfly parameterization

- The supports of all existing factors have the form:

$$\text{supp}(W^{(\ell)}) \subseteq \mathbf{I}_a \otimes \mathbf{1}_{b\times c} \otimes \mathbf{I}_d$$

EXAMPLE:

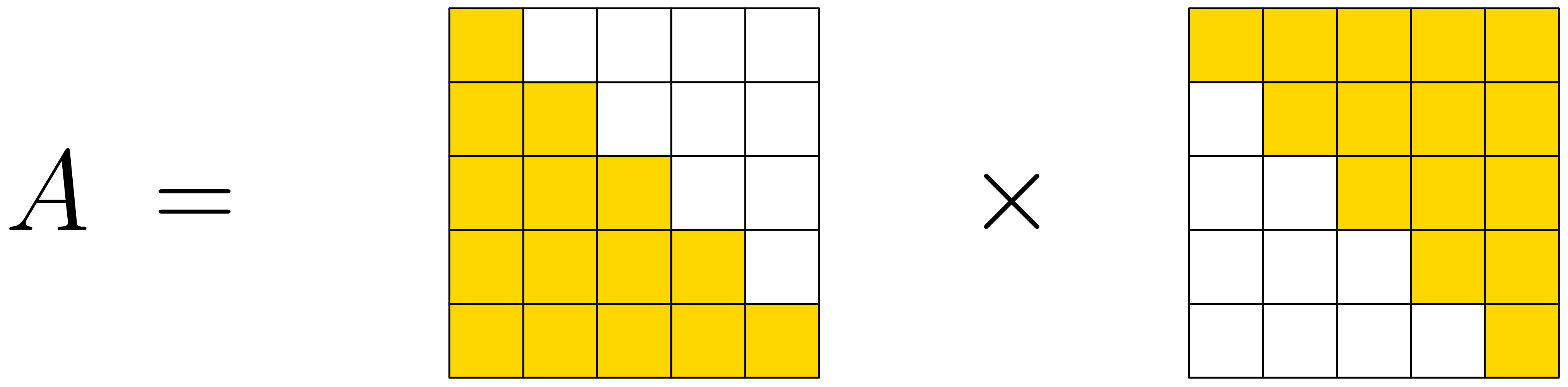
Parameterization	**Support forms**
Butterfly	$\mathbf{I}_{2^{\ell-1}} \otimes \mathbf{1}_{2\times 2} \otimes \mathbf{I}_{\frac{N}{2^\ell}}$
Kaleidoscope	
Monarch	$\mathbf{1}_{a\times b} \otimes \mathbf{I}_c$ and $\mathbf{I}_b \otimes \mathbf{1}_{c\times d}$
Deformable butterfly	$\mathbf{I}_a \otimes \mathbf{1}_{b\times c} \otimes \mathbf{I}_d$

- The product of two consecutive factors remains *butterfly* .

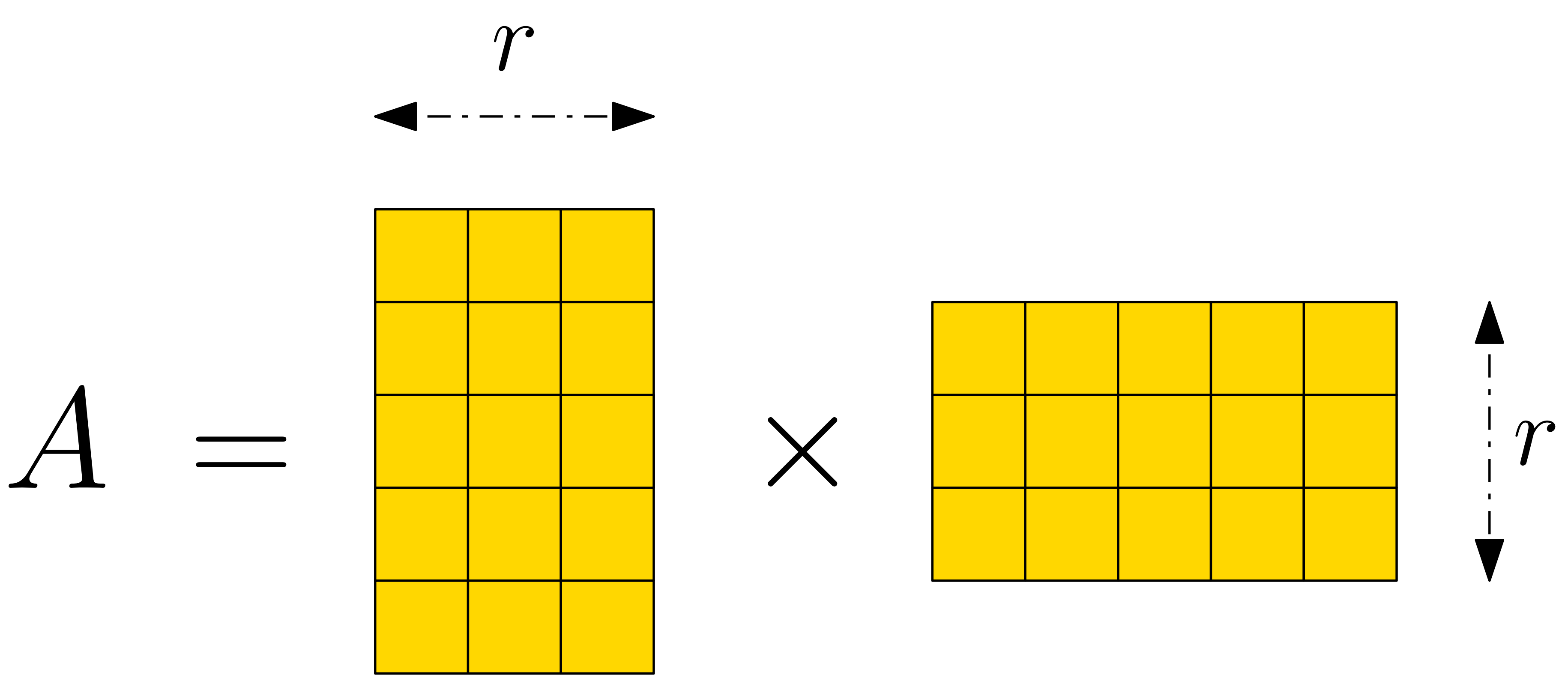
$$\text{supp}(W^{(\ell)}W^{(\ell+1)}) \subseteq \mathbf{I}_{a'} \otimes \mathbf{1}_{b'\times c'} \otimes \mathbf{I}_{d'}$$

⚠ This does not include the **Kaleidoscope** parameterization.

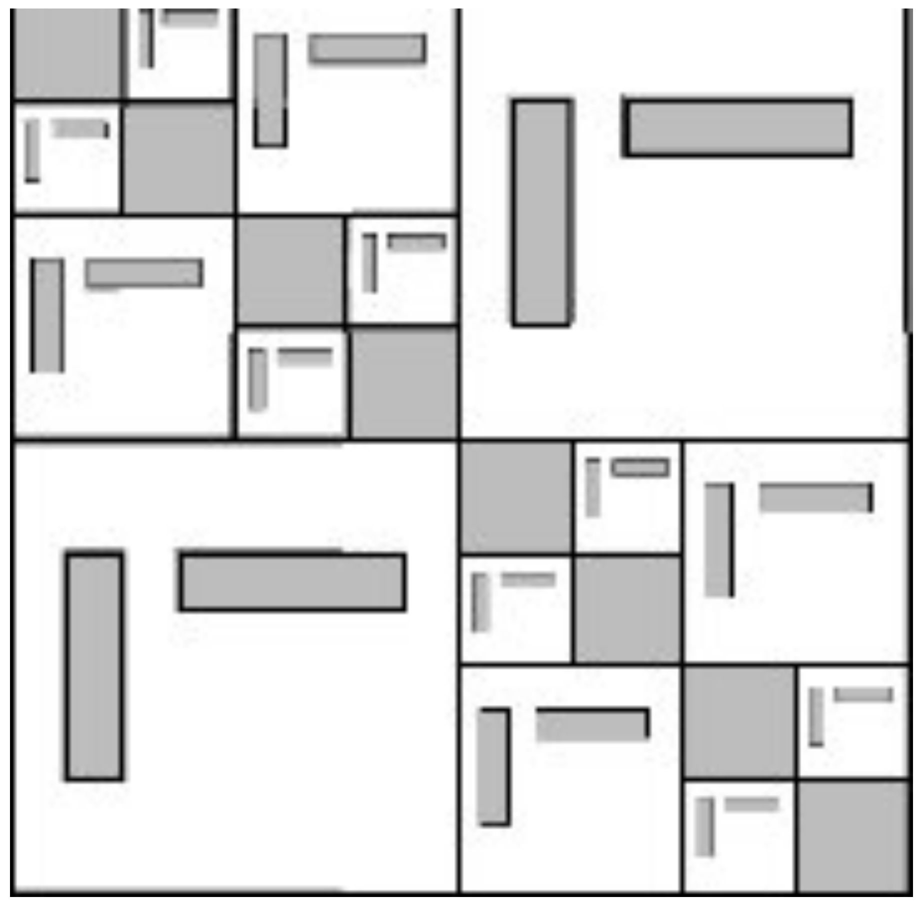
And Fixed Support Matrix Factorization



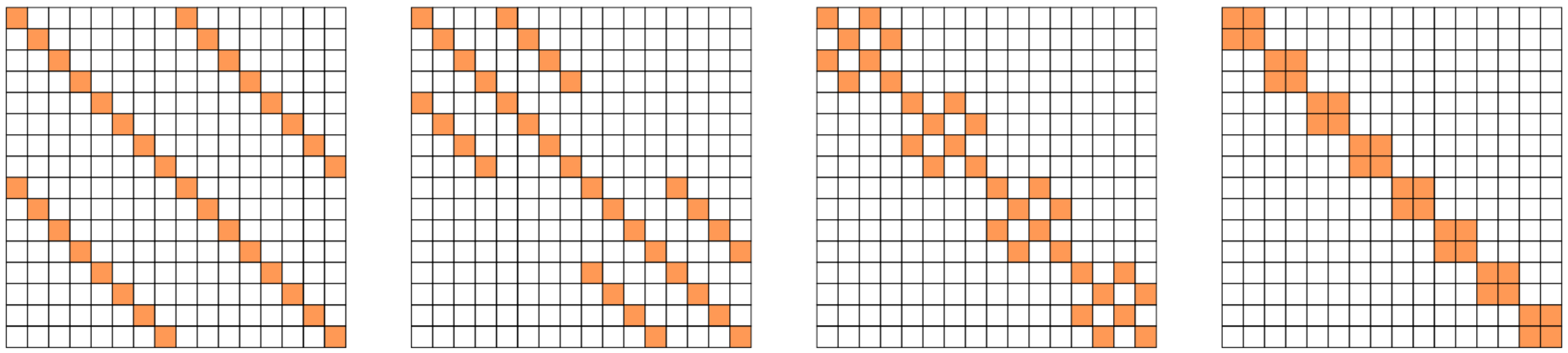
LU decomposition



Low rank approximation



Hierarchical matrix



(a) $\mathbf{S}_{\mathtt{bf}}^{(4)}$ (b) $\mathbf{S}_{\mathtt{bf}}^{(3)}$ (c) $\mathbf{S}_{\mathtt{bf}}^{(2)}$ (d) $\mathbf{S}_{\mathtt{bf}}^{(1)}$

Butterfly matrix factorization

Question for sparse neural networks

Problem formulation

Feed forward networks: $\theta = \{(\mathbf{W}_i, \mathbf{b}_i) \mid i = 1, \ldots, N\}$

$$f(x; \theta) = \mathbf{W}_N \sigma(\ldots \sigma(\mathbf{W}_1 x + \mathbf{b}_1)) + \mathbf{b}_N$$

Training:

Conventional Deep Neural Networks	Sparse Deep Neural Networks
Minimize$_\theta$: $\mathscr{L}_\theta := \sum_{i=1}^{n} L(f(\theta, x_i), y_i)$	Minimize$_\theta$: $\mathscr{L}_\theta := \sum_{i=1}^{n} L(f(\theta, x_i), y_i)$ such that: $\mathbf{W}_i$ are sparse matrices

Existing algorithms / approaches for Sparse Deep Neural Networks training:

- Pruning & Retraining, Lottery Ticket Hypothesis (Han et al., IPL 2015), (Zhu et al., 2017), (Jonathan et al., 2019)
- Regularisation l_0 or l_1 (Bengio et al., 2013), (Yu et al., 2017), (Collins et al., 2014), (Liu et al., 2015)
- Bayesian/ Variational approaches (Neklyudov et al., 2017), (Ullrich et al., 2017), (Louizos et al., 2017)