Tensor-based approaches for learning flexible neural networks

Yassine Zniyed
CRAN, Université de Lorraine, CNRS (Nancy)

Joint work with Konstantin Usevich, Sebastian Miron, David Brie

28.06.2021, Journée Théorie du DL

Table of Contents

What is a tensor ?

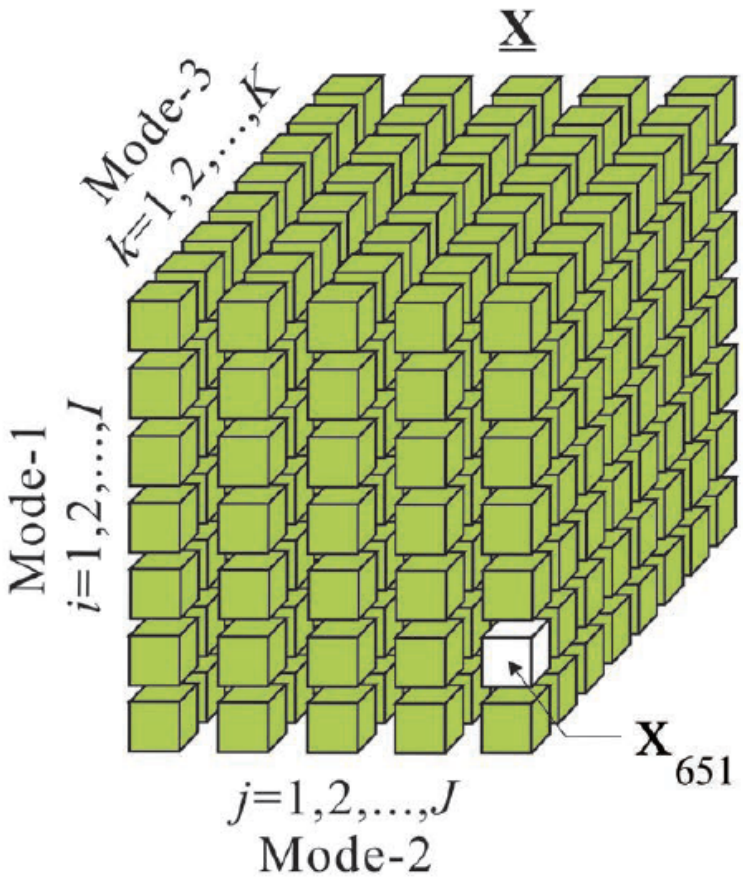


FIGURE – A 3-order tensor $\mathcal{X}$. [Cichocki et al., 2014]

Matrix and tensor rank-one decomposition

- $\circ$ outer product : $[\mathrm{p}_1 \circ \mathrm{p}_2 \circ \mathrm{p}_3]_{i,j,k} = \mathrm{p}_1(i) \cdot \mathrm{p}_2(j) \cdot \mathrm{p}_3(k)$.

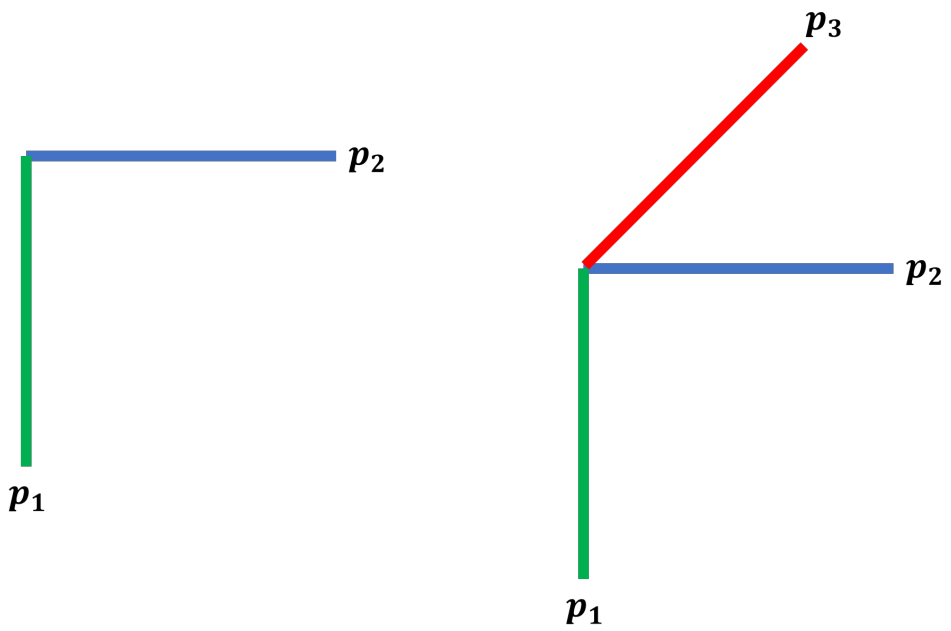


FIGURE – Left : Rank-1 matrix - Right : Rank-1 tensor.

- Matrix rank-1 decomposition : $\mathrm{X} = \mathrm{p}_1 \circ \mathrm{p}_2 = \mathrm{p}_1 \mathrm{p}_2^T$.
- What about rank-R decomposition ?

From matrix to tensor rank decomposition

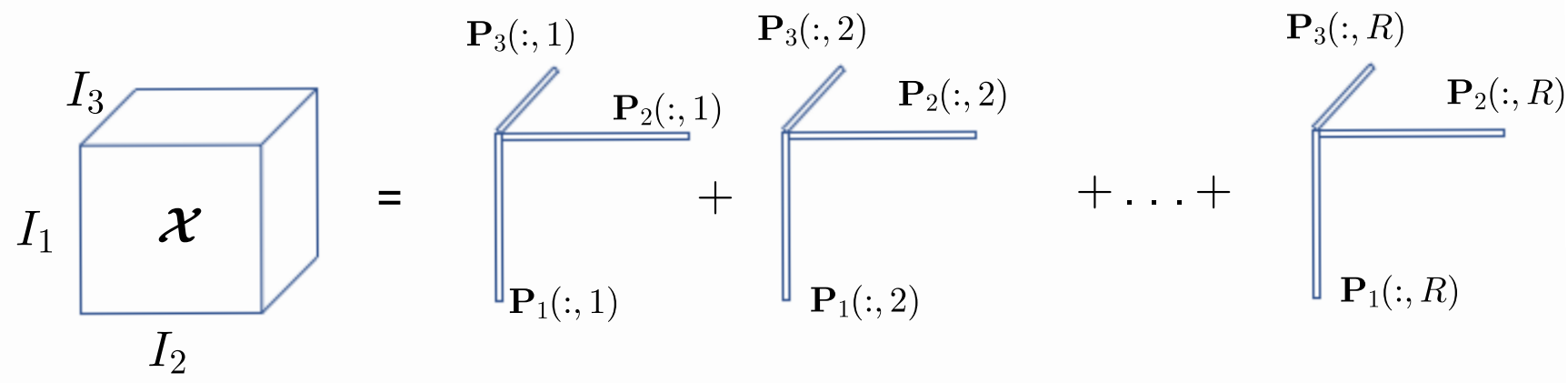


FIGURE – Tensor rank decomposition [Hitchcock, 1927].

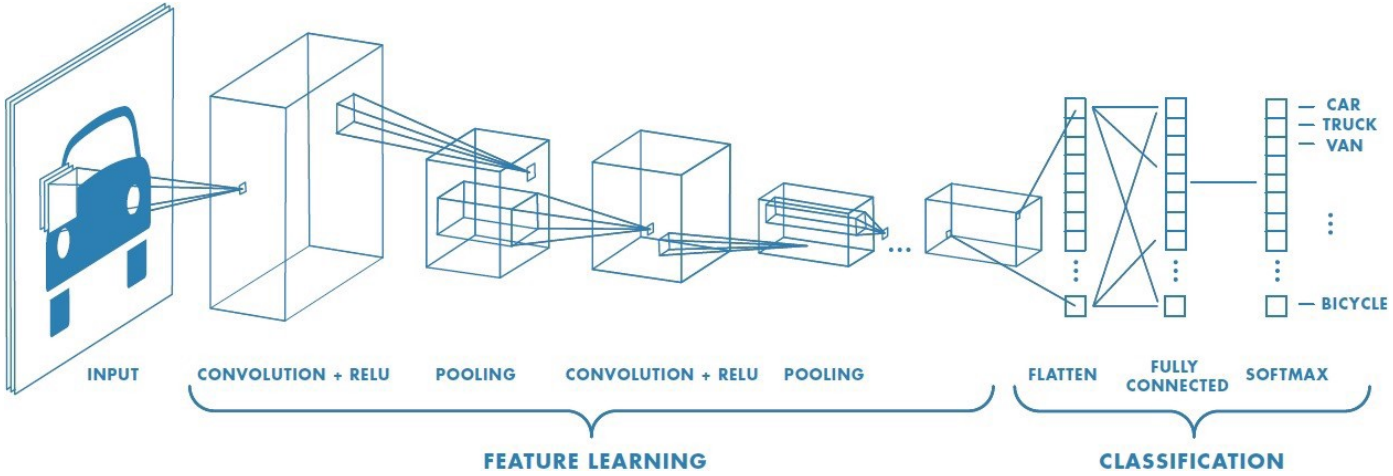
- Canonical polyadic decomposition (CPD) :

$$\mathcal{X} = \sum_{l=1}^{R} \mathrm{P}_1(:,l) \circ \mathrm{P}_2(:,l) \circ \mathrm{P}_3(:,l).$$

- Often unique, if rank R is smaller than a bound.

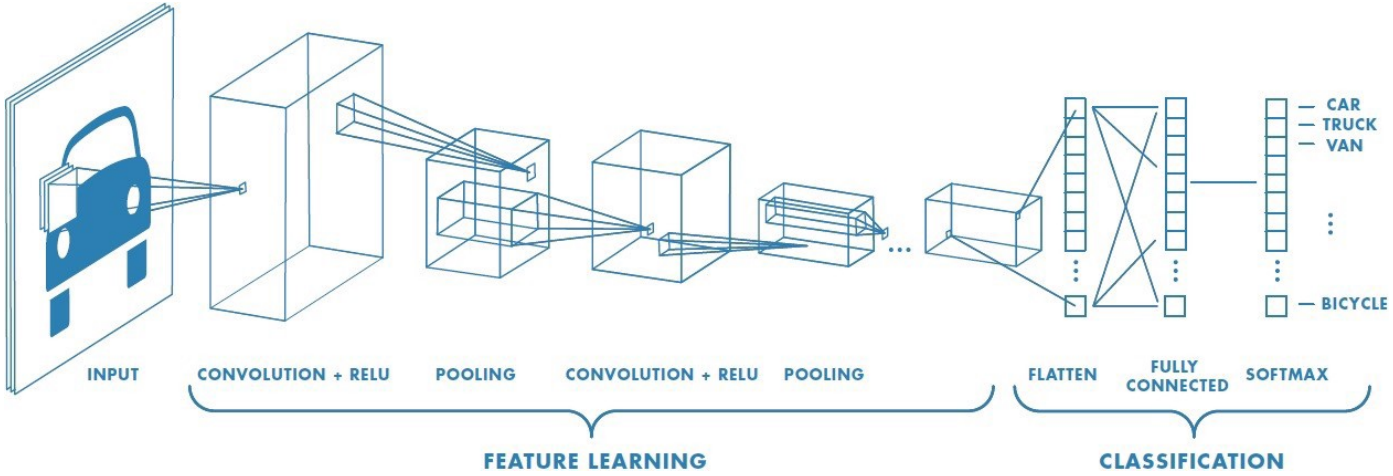
Overview

Typical feed-forward convolutional network :



Overview

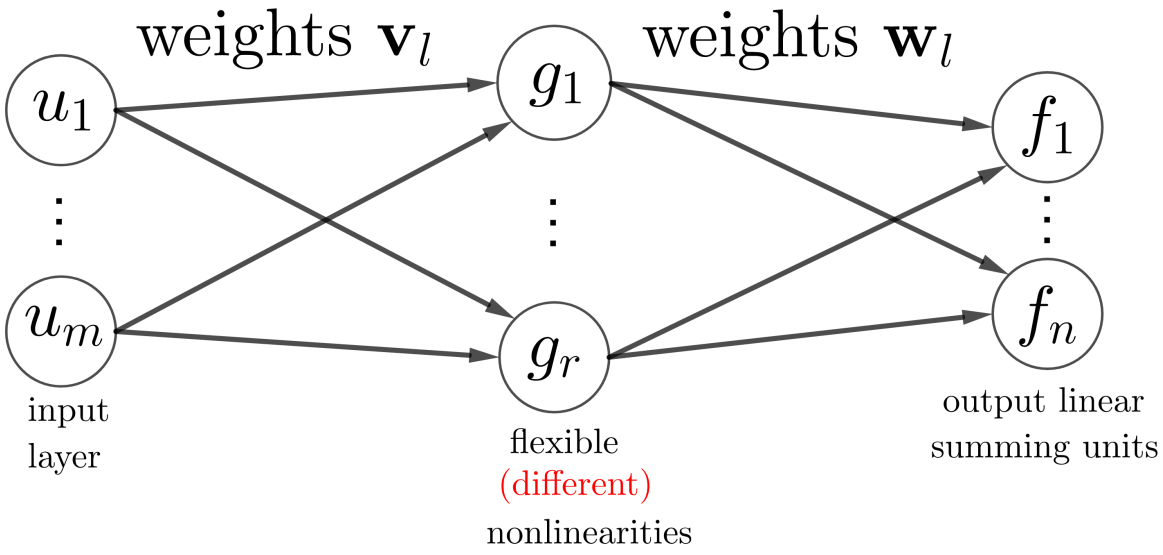
Typical feed-forward convolutional network :



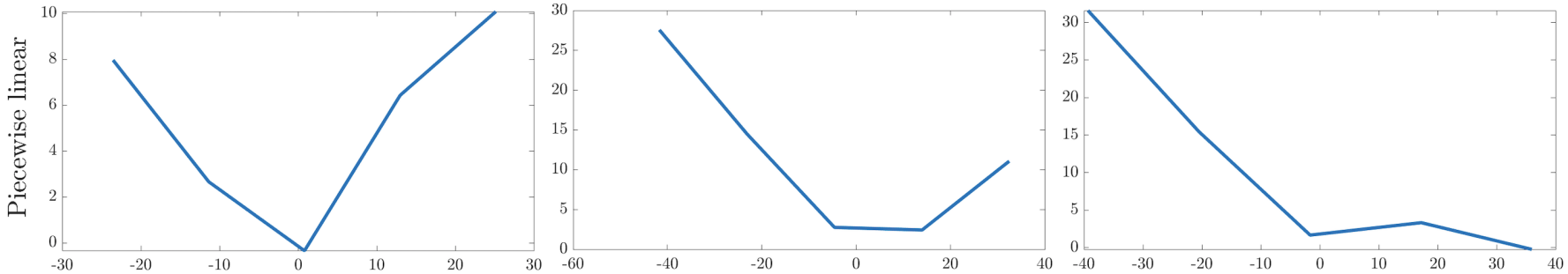
1. Compression of tensors in convolutional layers.
   Lebedev et al., "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition", ICLR, 2015.
2. Interpreting tensor decompositions as product-sum units.
   Cohen et al., "On the Expressive Power of Deep Learning : A Tensor Analysis", COLT, 2016
3. Our proposed architecture : flexible activation functions

Main idea

Graphical representation of the basic one-layer flexible NN :



Example :



Some motivation

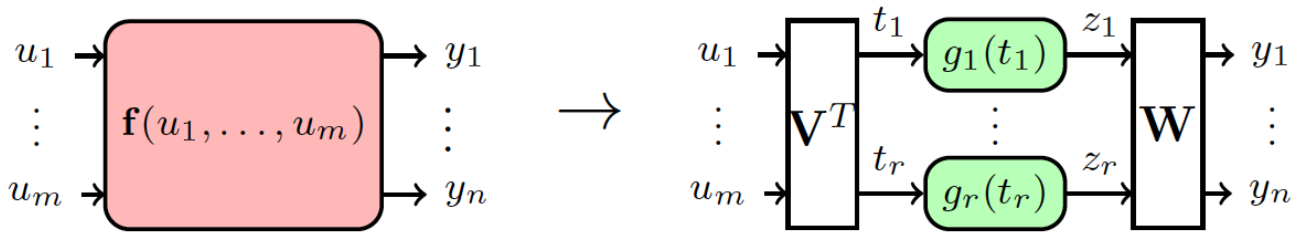
- Kolmogorov-Arnold representation theorem (1957) :
  Any continuous $f : [0,1]^m \to \mathbb{R}$ can be represented as

$$f(u) = \sum_{k=1}^{2m+1} \chi_k \Big(\sum_{j=1}^{m} \psi_{k,j}(u_j)\Big),$$

  with different continuous χ_k, $\psi_{k,j}$.
- Fixed AF with bias $(g(\mathrm{v}_k^T \mathrm{u} + b))$.
- Existing architectures (e.g. polynomial NN) are difficult to train from scratch.

Factorization : a tensor-based approach

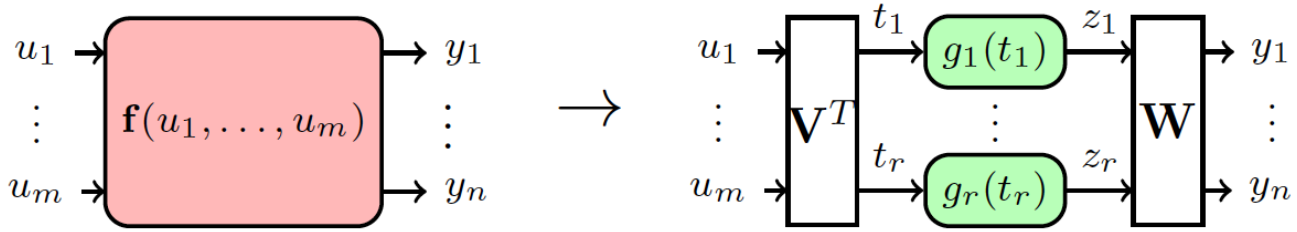
- Our framework : compression of pre-trained NN.
- Goal : decompose (approximate) given f by



i.e., $\mathrm{f}(\mathrm{u}) = \mathrm{W}\mathrm{g}(\mathrm{V}^T\mathrm{u})$, with $\mathrm{g}(t_1, \cdots, t_r) = [g_1(t_1) \cdots g_r(t_r)]^T$.

Factorization : a tensor-based approach

- Our framework : compression of pre-trained NN.
- Goal : decompose (approximate) given f by



i.e., $\mathrm{f}(\mathrm{u}) = \mathrm{W}\mathrm{g}(\mathrm{V}^T\mathrm{u})$, with $\mathrm{g}(t_1, \cdots, t_r) = [g_1(t_1) \cdots g_r(t_r)]^T$.

Borrow ideas from system identification ([Dreesen et al, 2015])

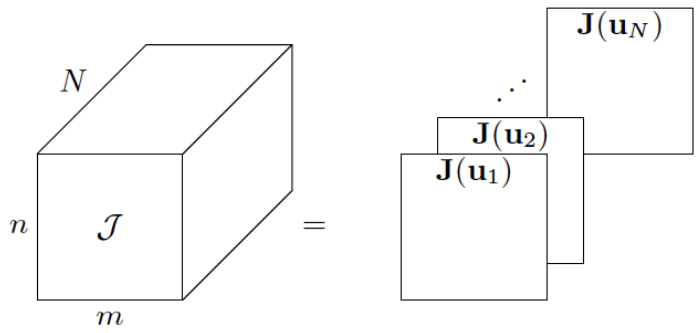
Key idea : by chain rule

$$\mathrm{J_f}(\mathrm{u}) := \begin{bmatrix} \frac{\partial f_1}{\partial u_1}(\mathrm{u}) & \cdots & \frac{\partial f_1}{\partial u_m}(\mathrm{u}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial u_1}(\mathrm{u}) & \cdots & \frac{\partial f_n}{\partial u_m}(\mathrm{u}) \end{bmatrix} = \mathrm{W} \cdot diag\left(g_1'(\mathrm{v}_1^T\mathrm{u}) \cdots g_r'(\mathrm{v}_r^T\mathrm{u})\right) \cdot \mathrm{V}^T.$$

Factorization : a tensor-based approach

Algorithm.

1. Evaluate $\mathrm{J_f}(\mathrm{u})$ at N "operating points" $\mathrm{u}_1, \ldots, \mathrm{u}_N \in \mathbb{R}^m$
2. Stack them into a tensor :



3. Joint matrix diagonalization $\leftrightarrow$ CPD

$$\begin{array}{ccc} \mathrm{J_f}(\mathrm{u}^{(1)}) = \mathrm{W}\mathrm{D}^{(1)}\mathrm{V}^\top, & & \\ \vdots & \leftrightarrow & \mathcal{J} = \sum_{l=1}^{r} \mathrm{w}_l \circ \mathrm{v}_l \circ \mathrm{h}_l \\ \mathrm{J_f}(\mathrm{u}^{(N)}) = \mathrm{W}\mathrm{D}^{(N)}\mathrm{V}^\top & & \end{array}$$

4. Retrieve v_l, w_l from factors of the CPD

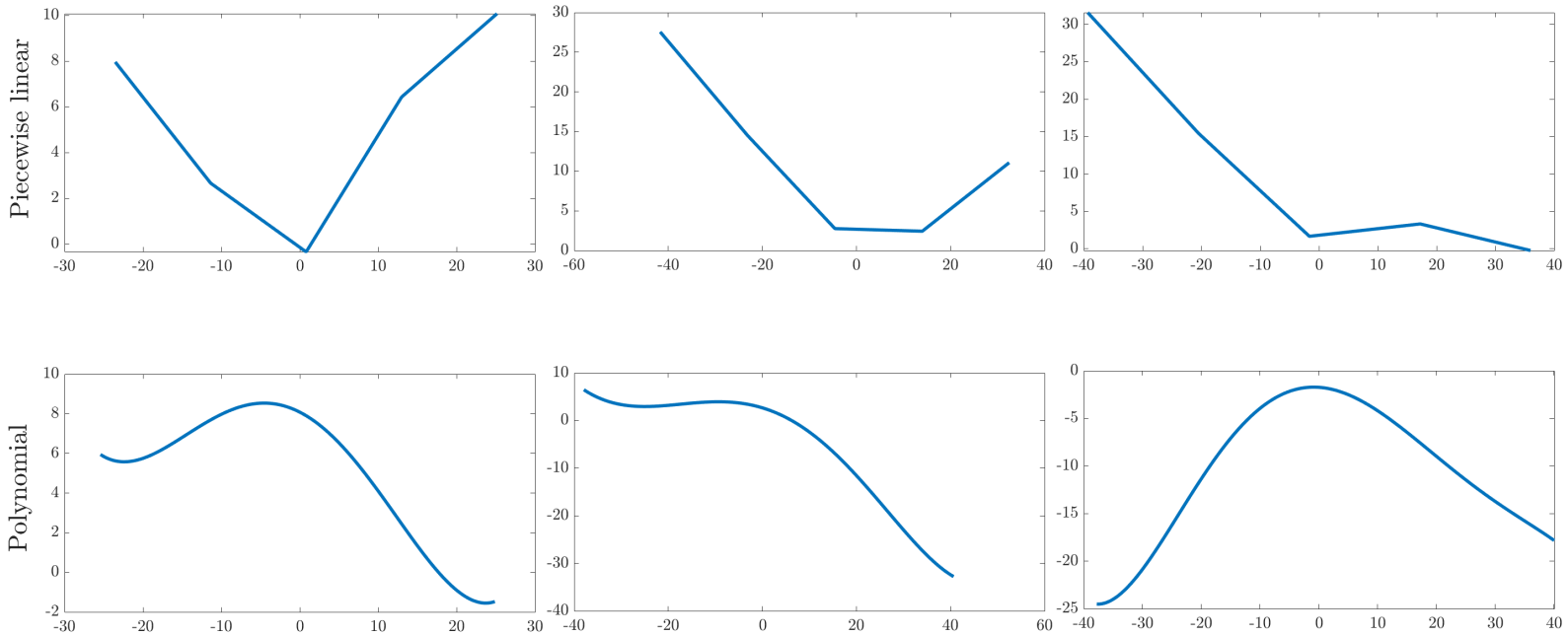
- Issue : assumptions needed to estimate g_l

Basis functions

The AFs are expressed as

$$g_l(t) = c_{0,l} + c_{1,l}\phi_1(t) + \cdots + c_{d,l}\phi_d(t).$$

Examples :



Uniqueness

The flexible NN model function $\mathrm{f} : \mathbb{R}^m \to \mathbb{R}^n$ is expressed as :

$$\mathrm{f}(\mathrm{u}) = \mathrm{w}_1 \cdot g_1(\mathrm{v}_1^T \mathrm{u}) + \cdots + \mathrm{w}_r \cdot g_r(\mathrm{v}_r^T \mathrm{u}). \tag{1}$$

Worth to mention : [Comon, Qi, Usevich, 2017]

- The above decomposition is associated with some interesting uniqueness results, when $\phi_k(t) = t^k$.
- The decomposition (1) is unique, when $d \geq 3$, $m \geq 2$ and r is not too large.
- The decomposition (1) is partially identifiable, when $r > mn$.

Constrained Coupled Matrix-Tensor Factorization (CMTF)

For a base $g_l(t) = c_{0,l} + c_{1,l}\phi_1(t) + \cdots + c_{d,l}\phi_d(t)$, we solve

$$\min_{\mathrm{w}_l,\mathrm{v}_l,\mathrm{h}_l,\mathrm{z}_l} \underbrace{\left\| \mathcal{J} - [|\mathrm{W},\mathrm{V},\mathrm{H}|] \right\|^2}_{\text{Jacobian}} + \lambda \cdot \underbrace{\left\| \mathrm{F} - \mathrm{W}\mathrm{Z}^T \right\|^2}_{\text{function evaluation (data fidelity)}}$$

$$\text{subject to} \quad \mathrm{h}_l = \mathrm{X}_l \cdot \mathrm{c}_l, \quad \mathrm{z}_l = \mathrm{Y}_l \cdot \mathrm{c}_l, \quad \text{with}$$

$$\mathrm{X}_l = \begin{bmatrix} 0 & \phi_1'(\mathrm{v}_l^T\mathrm{u}^{(1)}) & \cdots & \phi_d'(\mathrm{v}_l^T\mathrm{u}^{(1)}) \\ \vdots & \vdots & & \vdots \\ 0 & \phi_1'(\mathrm{v}_l^T\mathrm{u}^{(N)}) & \cdots & \phi_d'(\mathrm{v}_l^T\mathrm{u}^{(N)}) \end{bmatrix}, \quad \mathrm{Y}_l = \begin{bmatrix} 1 & \phi_1(\mathrm{v}_l^T\mathrm{u}^{(1)}) & \cdots & \phi_d(\mathrm{v}_l^T\mathrm{u}^{(1)}) \\ \vdots & \vdots & & \vdots \\ 1 & \phi_1(\mathrm{v}_l^T\mathrm{u}^{(N)}) & \cdots & \phi_d(\mathrm{v}_l^T\mathrm{u}^{(N)}) \end{bmatrix}.$$

- We use an alternating least squares (block-coordinate descent) with projection

Neural networks compression

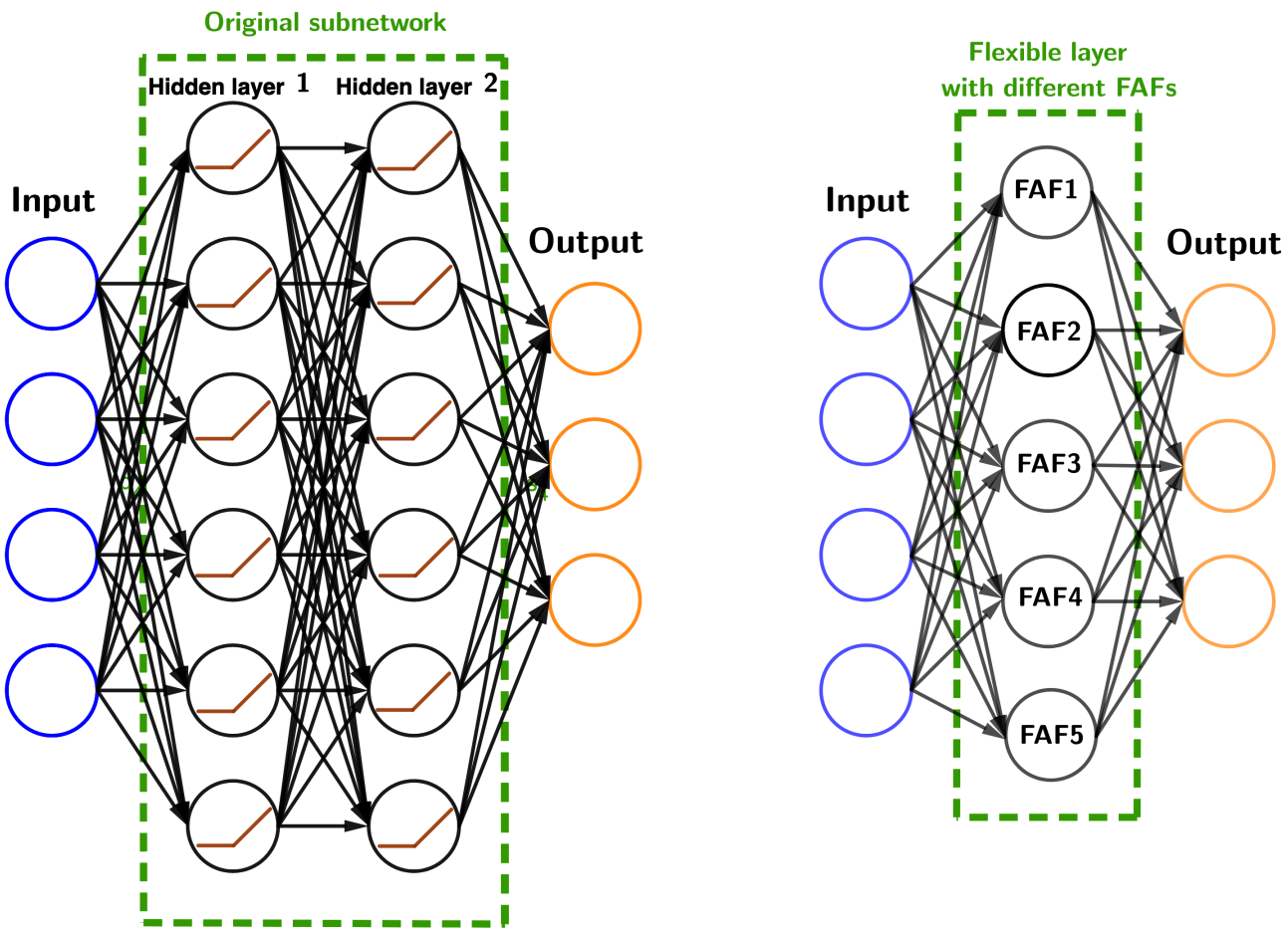
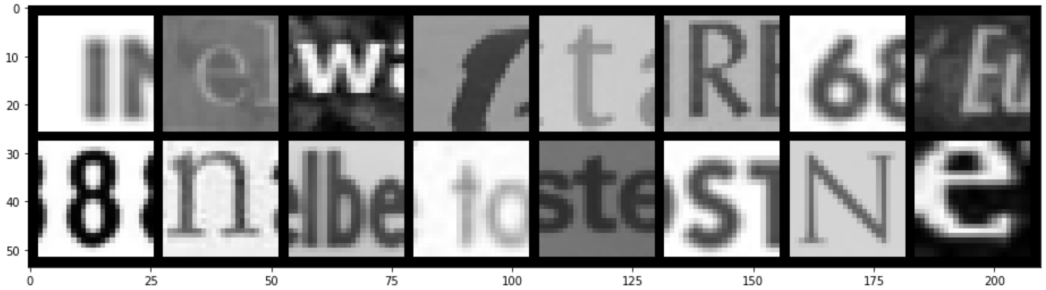


FIGURE – Graphical representation of the compression process : (left) the original pretrained NN with fixed AFs - (right) the approximated NN with flexible AFs.

ICDAR and CharNet

ICDAR 2003 dataset : ≈163000 train, ≈ 5300 test, 36 classes

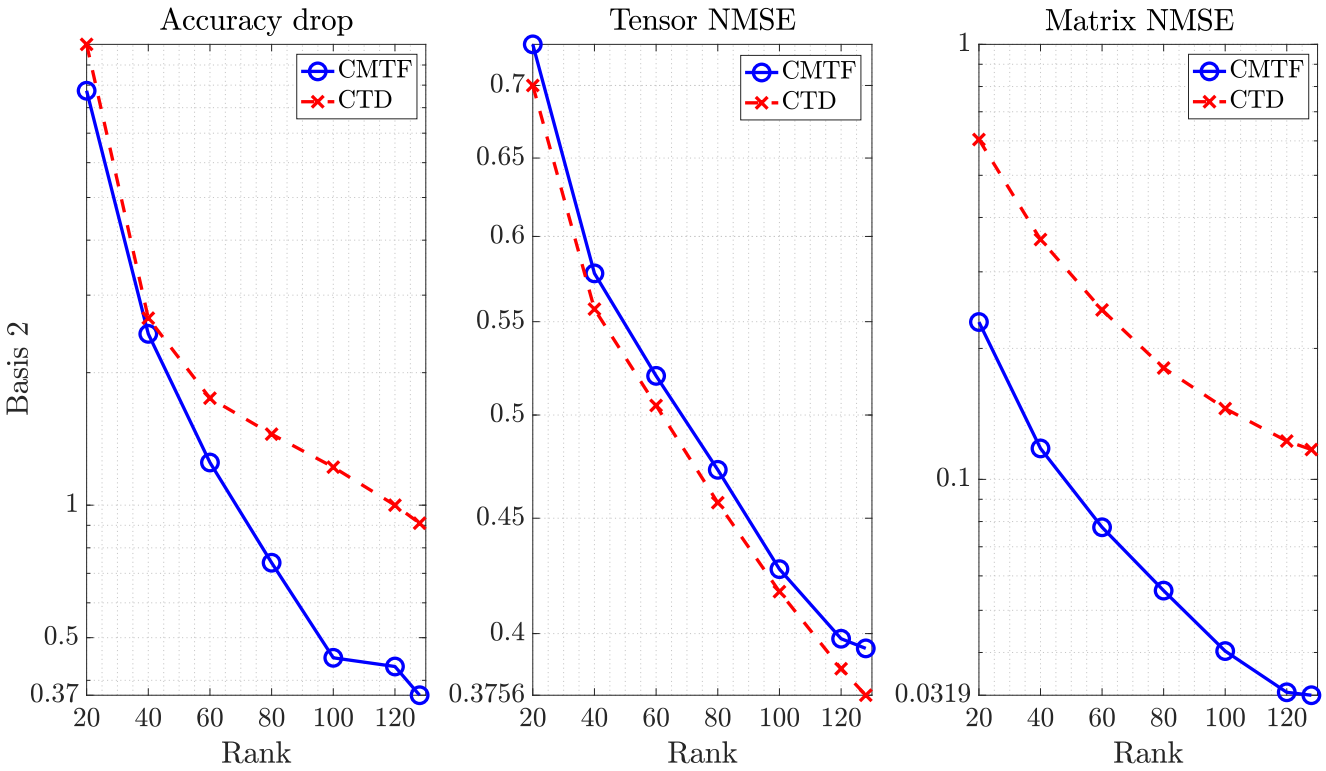


CharNet [Jaderberg et al., 2014] (in MatConvNet from Oxford)

layer	type	filter size	input	output
conv1	convolutional	9x9	1	96
	maxout (/2)		96	48
conv2	convolutional	9x9	48	128
	maxout (/2)		128	64
conv3	convolutional	8x 8	64	512
	maxout (/4)		512	128
fc	fully connected	1x 1	128	192
	maxout (/4)		192	48
softmax	softmax	-	48	36

Some results

- we compress the conv3 layer of CharNet (viewed as fully connected layer $\mathbb{R}^{4096} \to \mathbb{R}^{128}$)
- use only 360 points (10 per class), without fine-tuning
- 4x compression



Thank you !

References :

- P. Comon, Y. Qi and K. Usevich, *X-rank and identifiability for a polynomial decomposition model*, SIAGA, 2017.
- K. Usevich, P. Dreesen and M. Ishteva., *Decoupling multivariate polynomials : interconnections between tensorizations*, JCAM, 2020.
- Y.Zniyed, K. Usevich, S. Miron, D. Brie, *Tensor-based framework for training flexible neural networks*, preprint on arXiv :2106.13542.