Improving AI with random tensors

Mohamed Ouerfelli

CEA List

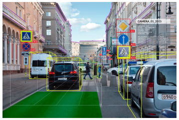
Quantum Gravity in Bordeaux, July 7th, 2023

Based on ongoing work with Parham Radpay, Mohamed Tamaazousti and Vincent Rivasseau
and on ongoing work with Théo Rudkiewicz and Mohamed Tamaazousti

Overview

Artificial intelligence applications

Existent applications of Artificial intelligence



Computer vision

NLP

Health data

Autonomous driving

Quantum Gravity in Bordeaux 2023

Random Tensors, from Quantum Gravity to Artificial Intelligence

July 5-7, 2023
LaBRI, Campus Peixotto, Talence, France

Organisers:
Adrian Tanasa, *université de Bordeaux, France*
Thomas Muller, *université de Bordeaux, France*

Invited speakers:
Juan Abranches, *Okinawa Institute of Science and Technology, Japan*
Dario Benedetti, *École polytechnique, France*
Joseph Ben Geloun, *Université Sorbonne Paris Nord, France*
Sylvain Carrozza, *Université de Bourgogne, Dijon, France*
Thomas Krajewski, *Aix-Marseille Université, France*
Davide Lettera, *Universität Heidelberg, Germany*
Luca Lionni, *Universität Heidelberg, Germany*
Mohamed Ouerfelli, *CEA List, France*

cnrs LaBRI université BORDEAUX

Image Generation

Robotics

AI Methods

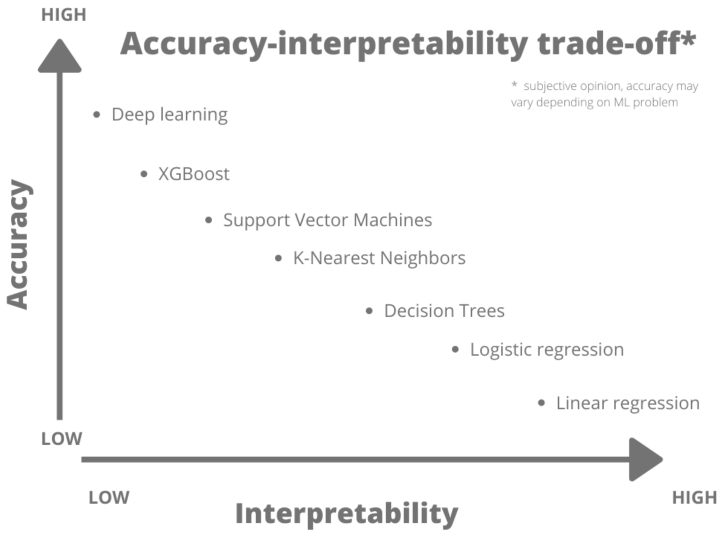


Figure: AI Methods

AI Methods

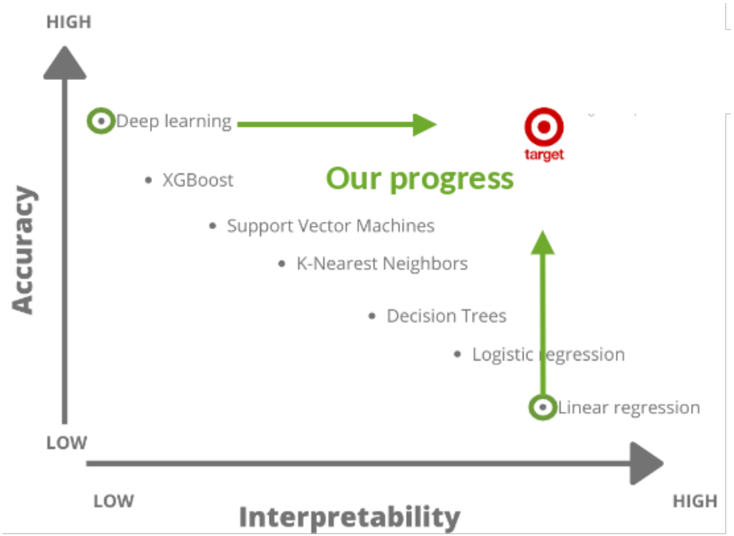
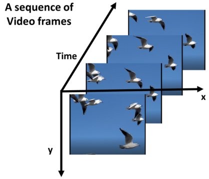


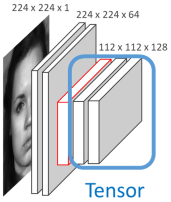
Figure: AI Methods

Tensorial data

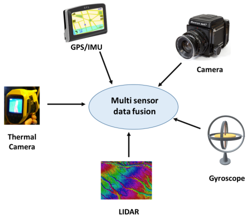
- Powerful computers and acquisition devices have made it possible to capture and store real-world multidimensional data.



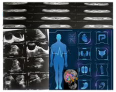
Video data



Neural network layer

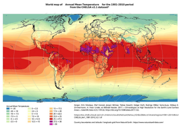


Multi sensor data fusion

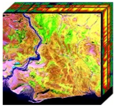


Unstructured Personal health data set

Health data



Weather data



Hyperspectral images

Figure: Examples of tensorial data

Tensor Principal Component Analysis (PCA)

- Tensor PCA is a statistical model that consists in inferring an unknown unit vector v_0 from a tensor $\mathbf{T} \in (\mathbb{R}^n)^{\otimes k}$ given by:

$$\mathbf{T}_{i_1\dots i_k} = \sqrt{n}\ \underset{\substack{\nearrow\\ \text{Signal-to-noise ratio}}}{\beta}\ \underset{\substack{\nwarrow\\ \text{Signal}}}{\mathbf{v}_{i_1}\otimes\cdots\otimes\mathbf{v}_{i_k}} + \underset{\substack{\uparrow\\ \text{Noise}}}{\mathbf{Z}_{i_1\dots i_k}}$$

with Z Gaussian noise tensor such that $Z_{i_1\dots i_k} \sim \mathcal{N}(0,1)$ and β the **signal-to-noise ratio** (SNR).

Trace invariants definition

- A tensor T transforms under the group $\bigotimes_{a=1}^{k} O(n)$ as:

$$\mathbf{T}_{a_j^1 \dots a_j^k} \to O^{(1)}_{a_j^1 b_j^1} \dots O^{(k)}_{a_j^k b_j^k} \mathbf{T}_{b_j^1 \dots b_j^k} \quad \text{for} \;\; O^{(i)} \in O(n) \;\; \forall i \in [k] \qquad (1)$$

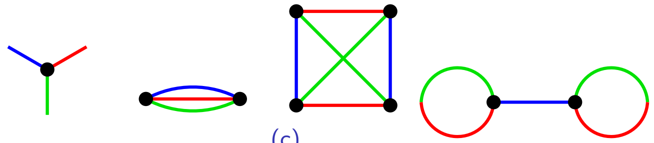
- Contraction of two indices : making them equal and summing over them. Einstein notation : two identical indices indicates an implicit summation over them.

- Example: a trace of a matrix consists in contracting its two indices.

$$\mathrm{Tr}(\boldsymbol{M}) = \sum_i \boldsymbol{M}_{ii} \underset{\substack{\big\uparrow \\ \text{Einstein notation}}}{\equiv} \boldsymbol{M}_{ii}$$

Graphs associated to Trace invariants

- Trace invariants are scalars obtained by contracting $2n$ copies of $\mathbf{T}$ invariant under $\bigotimes_{a=1}^{k} O(n)$. They have a very simple illustration as graphs.



(a) T_{ijk} (b) $T_{ijk}T_{ijk}$ (c) $T_{ijk}T_{ij'k'}T_{i'jk'}T_{i'j'k}$ (d) $T_{ijj}T_{ikk}$

Figure: Example of graphs and their associated invariants

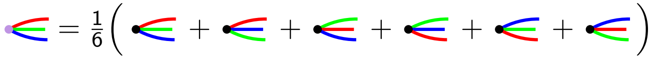
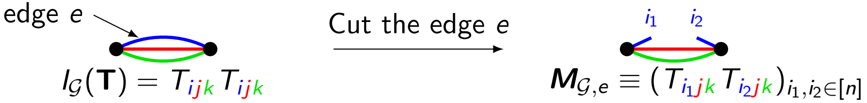


Figure: Depiction of symmetrization as the averaged sum of the permutations. Note that in the left hand side the order of edges does not matter since symmetric tensors are by definition invariant under permutation of indices.

Matrices associated to a graph

- An invariant should be able to detect a signal. But we want matrices to recover it.
- To this effect, we introduce a new set of tools in the form of matrices. We denote by $\boldsymbol{M}_{\mathcal{G},e}$ the matrix obtained by cutting an edge e of a graph $\mathcal{G}$ in two half edges.



$$I_{\mathcal{G}}(\mathbf{T}) = T_{ijk}T_{ijk} \qquad \boldsymbol{M}_{\mathcal{G},e} \equiv (T_{i_1jk}T_{i_2jk})_{i_1,i_2\in[n]}$$

Figure: Obtaining a matrix by cutting the edge of a trace invariant graph $\mathcal{G}$

The algorithm

Algorithm 1: Recovery algorithm associated to the graph $\mathcal{G}$ and edge e

```
**Input:** The tensor $\mathbf{T} = \beta \boldsymbol{v}^{\otimes k} + \mathbf{Z}$
**Goal:** Estimate $\boldsymbol{v}_0$.
Calculate the matrix $\boldsymbol{M}_{\mathcal{G},e}(\mathbf{T})$
Compute its top eigenvector
**Output:** Obtaining an estimated vector $\boldsymbol{v}$
```

SOTA

- It appears that the two state of the art (SOTA) methods are equivalent to the algorithms associated to graphs of degree 2.
- This striking fact incites us to investigate the algorithm associated to the tetrahedral graph which is a graph of degree 4 as illustrated in the following Figure.

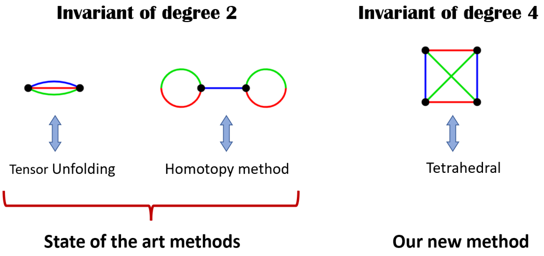


Figure: Methods associated to invariant graphs

Numerical experiment

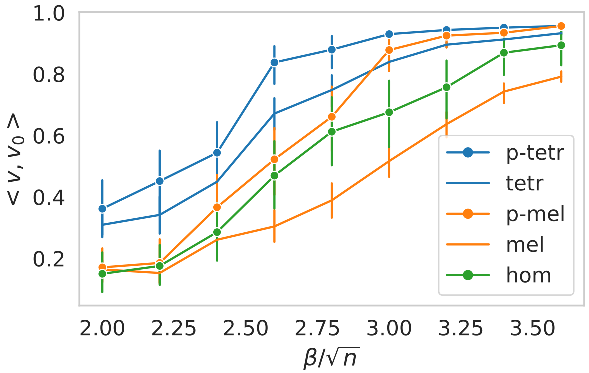


Figure: Comparison of Tetrahedral (blue) with the two SOTA methods (orange and green) for n=150.

Signal detection

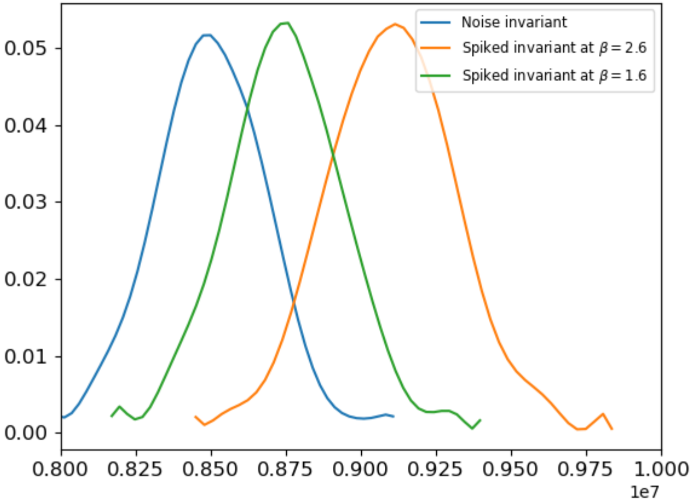


Figure: Shift of invariant distribution as β increases in this case from 0 to 2.6 for

Sum of graphs

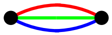
- The state-of-the-art algorithms for spiked tensor models have been translated into certain categories of trace invariants, and some improvements have been suggested. However, the combination of these graphs has not yet been explored.
- In the following, we will introduce the vector space spanned by certain categories of algorithms from RTT and study them in the particular case of signal detection.

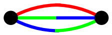
Categories of graphs

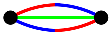
- In the case of symmetric tensors, under the constraint that the degree of the invariant is equal to or less than 4, one can only generate four possible trace invariants.
- These trace invariants are depicted by *melonic, tadpole, tetrahedral* and *pillow* graphs.
- In the case of non-symmetric tensors, each of these invariants, generates a different family of a different number of invariants. But each one of these four families is closed under permutation of the indices of the tensor.

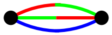
Melonic category

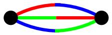
- **Melonic Graphs** build a category of trace invariants that describe a contraction between two copies of a tensor.
- Considering figure 6, thus one can naively assume that there are 36 possible contractions. However, there are some redundancies that can be factored out, thus decreasing the number of invariants from 36 to 6. The 6 graphs can be seen in figure 11.


(a) $I_1 = T_{ijk} T_{ijk}$


(b) $I_2 = T_{ijk} T_{ikj}$


(c) $I_3 = T_{ijk} T_{kji}$


(d) $I_4 = T_{ijk} T_{jik}$


(e) $I_5 = T_{ijk} T_{jki}$

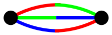
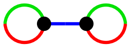
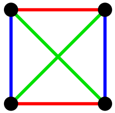

(f) $I_6 = T_{ijk} T_{kij}$

Figure: 6 possible melonic graphs resulting from permuting the half-edges on the right vertex. However, a sequence of renaming of indices $i \leftrightarrow k$ and $j \leftrightarrow k$, which is equivalent to exchanging colors in the graph, shows that $I_5 = I_6$

Tadpole, Tetrahedral and pillow graphs

- The next category of invariants are **tadpole graphs**. There are overall 6 possible graphs in this category, as shown in the supplementary material.
- Next, the **tetrahedral graphs**. These construct the class of complete graphs with 4 vertices. This means algebraically that there will be a contraction between all copies of the tensor.
- Finally, the last category defines the **pillow graphs**, which are graphs of degree 4 constructed by double contraction between two pairs of tensors.


(a) $I_1 = T_{jjk} T_{iik}$


(b) $T_{ijk} T_{ij'k'} T_{i'jk'} T_{i'j'k}$

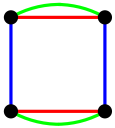

(c) $T_{ijk} T_{i'jk} T_{ij'k'} T_{i'j'k'}$

Figure: Examples of tadpole, tetrahedral and pillow graphs.

Loss function

By defining a function with the distance of the means in the numerator and the sum of the standard deviations in the denominator, we obtain a function whose maximum would give us the largest *separation* of the two graphs. This quantity is given by:

$$f_\beta(I) = \frac{m_S(I,\beta) - m_N(I)}{\sigma_N(I) + \sigma_S(I,\beta)} \tag{2}$$

where $m_N(I)$ and $\sigma_N(I)$ are the mean and standard deviation of the trace invariant, I, of a pure noise tensor and $m_S(I,\beta)$ and $\sigma_S(I,\beta)$ correspond to the mean and standard deviation of the same invariant in the case of a spiked tensor with the signal to noise ratio, β. The numerator in this expression is basically the contribution of the signal to the mean of the invariant.

Simplification and requirements

For the invariants that fulfil some conditions we can minimize the following quantity to optimize the algorithm:

$$\frac{\sigma_N\left(\sum_i \alpha_i I_i\right)}{\sum_i \alpha_i}$$

Assuming the condition that $\mathrm{Cov}(I_i, I_j) \ll \mathrm{Var}(I_i), \mathrm{Var}(I_j)$ for each I_i and I_j under the family of the considered invariants, elementary calculations in the appendix show that this is minimized at:

$$\alpha_i = \mathcal{N}\frac{1}{\mathrm{Var}_i} \tag{3}$$

where $\mathcal{N}$ is a normalization factor that is constant for every factor.

Experimental results

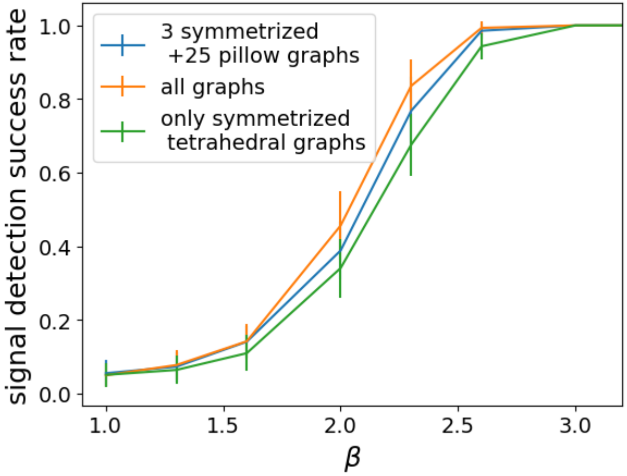


Figure: Signal detection for three different algorithms.

Deep learning limitations

Neural networks suffer from the following limitations :

- Overparametrized
- High energy cost
- A poor understanding of how they work, "black boxes"

Neural network

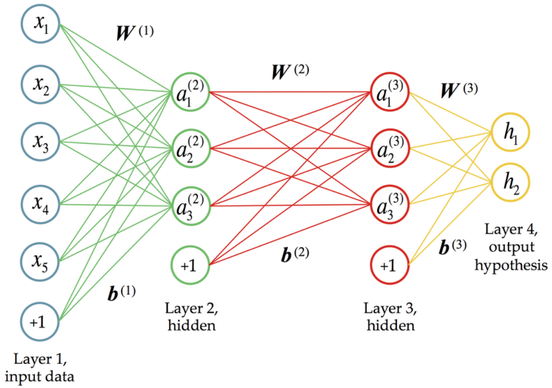


Figure: Neural network schema (image from [Sheeran2016])

Linear layer and SVD

- A linear layer is a function $f : \mathbb{R}^n \to \mathbb{R}^m$ defined by $f(x) = Mx$ where $M \in \mathbb{R}^{m \times n}$.
- There are different methods to approximate M with a low rank, the most popular method is based on singular value decomposition $U\Sigma V = M$
- To compress M with a small loss of information we can choose to neglect the small singular values and keep only the r largest singular values. We then get:

$$\widetilde{M} = \sum_{i=1}^{r} \sigma_i U_i V_i^\top$$

CNN and tensors

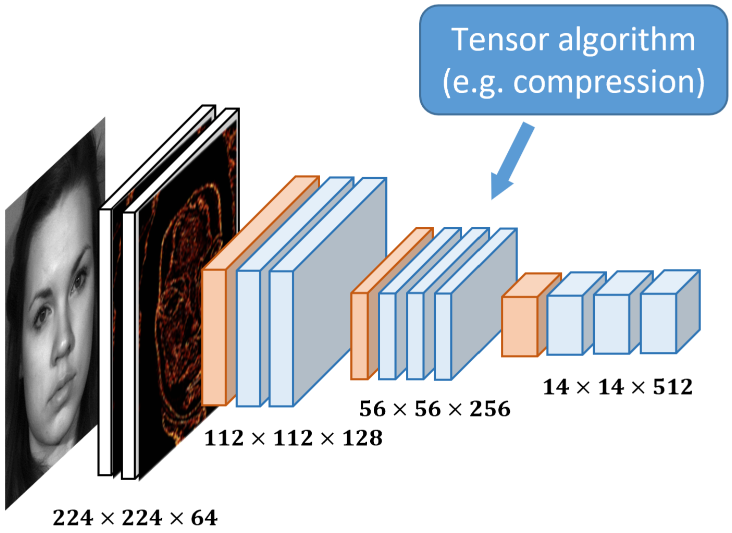

Tensor algorithm (e.g. compression)
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$14 \times 14 \times 512$

Figure: Tensors in a neural network

Tensor decomposition

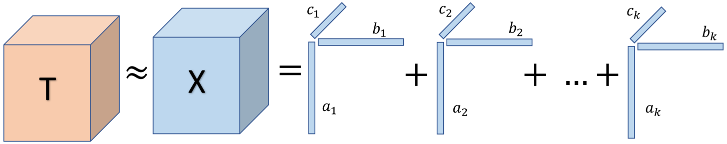


Figure: CP decomposition $\mathbf{T} \equiv \sum_{i=1}^{r} \lambda_i (\boldsymbol{v}_1^{(i)} \otimes \dots \boldsymbol{v}_k^{(i)})$

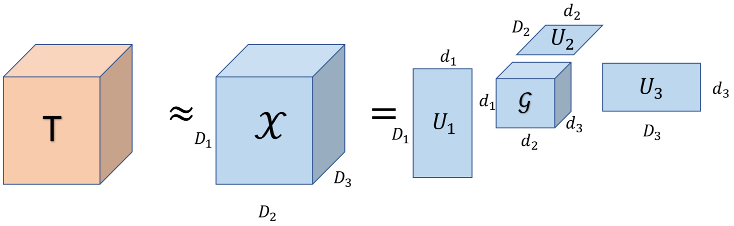


Figure: Tucker decomposition $\mathbf{T} = \mathcal{G} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3$

CP decomposition

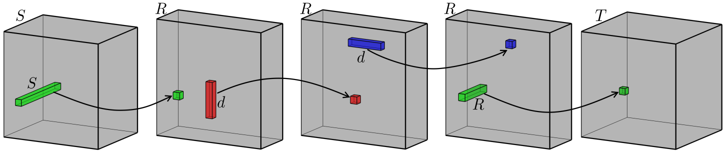


Figure: CP convolution

$$\mathcal{Y}[t,y,x] = \sum_{r=1}^{R} \sum_{w=-w_d}^{w_d} \sum_{h=-h_d}^{h_d} \sum_{s=1}^{S} U_r^{(T)}[t] U_r^{(W)}[w] U_r^{(H)}[h] U_r^{(S)}[s] \mathcal{X}[s, y+h, x+w]$$

$$= \underbrace{\sum_{r=1}^{R} U_r^{(T)}[t] \underbrace{\left[\sum_{w=-w_d}^{w_d} U_r^{(W)}[w] \underbrace{\left(\sum_{h=-h_d}^{h_d} U_r^{(H)}[h] \underbrace{\left[\sum_{s=1}^{S} U_r^{(S)}[s] \mathcal{X}[s, y+h, x+w] \right]}_{1\times 1 \text{ conv}} \right)}_{\text{depthwise conv}} \right]}_{\text{depthwise conv}}}_{1\times 1 \text{ convolution}}$$

Tucker decomposition

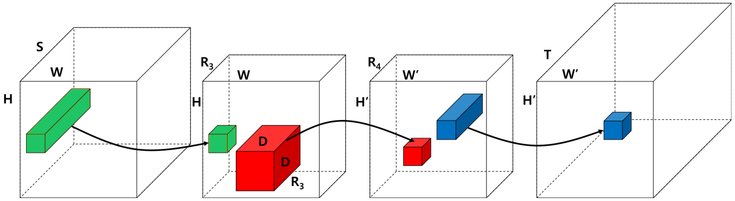


Figure: Tucker convolution

$$\mathcal{Y}[t,y,x] = \sum_{s=1}^{S}\sum_{h=-h_d}^{h_d}\sum_{w=-w_d}^{w_d}\sum_{r_s=1}^{R_S}\sum_{r_t=1}^{R_T} \mathcal{G}[r_t,r_s,h,w]U^{(T)}[r_t,t]U^{(S)}[r_s,s]\mathcal{X}[s,y+h,x+w]$$

$$= \underbrace{\sum_{r_t=1}^{R_T} U^{(T)}[t,r_t]\left[\underbrace{\sum_{h=-h_d}^{h_d}\sum_{w=-w_d}^{w_d}\sum_{r_s=1}^{R_S}\mathcal{G}[r_t,r_s,h,w]\left[\underbrace{\sum_{s=1}^{S}U^{(S)}[s,r_s]\mathcal{X}[s,h+y,w+x]}_{1\times 1\text{ conv}}\right]}_{H\times W\text{ conv}}\right]}_{1\times 1\text{ conv}}$$

What criteria

- The best way to compute the the decomposition is not trivial. Standard algorithms focus on minimizing a classic norm like L_1, L_∞ or in most of the case L_2.

Sigma norm

Prior knowledge about the distribution of the input x distribution can be taken into account by considering the weighted approximation problem as suggested by [Denton et al., 2014].

Proposition

If we suppose that the distribution of the input x follows a distribution $\mathcal{L}$ such that $\forall i, i, \operatorname{cov}(x[i], x[j])$ is well defined, we have

$$\mathbb{E}\left|\mathcal{K}_{(1)}x - \widetilde{\mathcal{K}}_{(1)}x\right| = \left\|\left(\mathcal{K} - \widetilde{\mathcal{K}}\right)_{(1)} \Sigma^{1/2}\right\|_F \tag{4}$$

Where $\Sigma^{1/2}(\Sigma^{1/2})^\top = \mathbb{E}xx^\top$

Taking into account the noise

- We are minimizing $\left\|\mathcal{K} - \tilde{\mathcal{K}}\right\|$
- If we consider that $\mathcal{K} = \mathcal{S} + \mathcal{Z}$ where $\mathcal{Z}$ is noise. We wish to minimize $\left\|\mathcal{S} - \tilde{\mathcal{K}}\right\|$ instead of $\left\|\mathcal{K} - \tilde{\mathcal{K}}\right\|$

Choice of the rank

To automatically choose a good rank we use the VBMF (Variational Bayesian Matrix Factorization) algorithm [Nakajima et al., 2010] on a matrix obtained by flattening one of the axis of the tensor.
The VBMF is an algorithm to estimate the rank of a matrix. We suppose that we have to a noisy version of the matrix (precisely we have access to the matrix plus a gaussian noise matrix $\boldsymbol{Z}$).

$$\boldsymbol{M} = \boldsymbol{S} + \boldsymbol{Z}$$

The VBMF algorithm then estimate the rank of the underlying matrix $\boldsymbol{S}$.
We are working on a method directly on the tensor using our previous framework and Random Tensor Theory.

Robustness

- Neural networks achieve state of the art performances on pure accuracy test.
- However, they are known to be vulnerable to slight perturbations of the input known as adversarial attacks.
- Those attacks are a major concern for the deployment of neural networks in real world applications.
- In the case of compressed neural networks, the primary goal is to make them available on devices with limited resources to use them in real applications such as autonomous cars or drones. Hence, it is important to study the robustness of compressed neural networks to adversarial attacks.

Adversial attacks

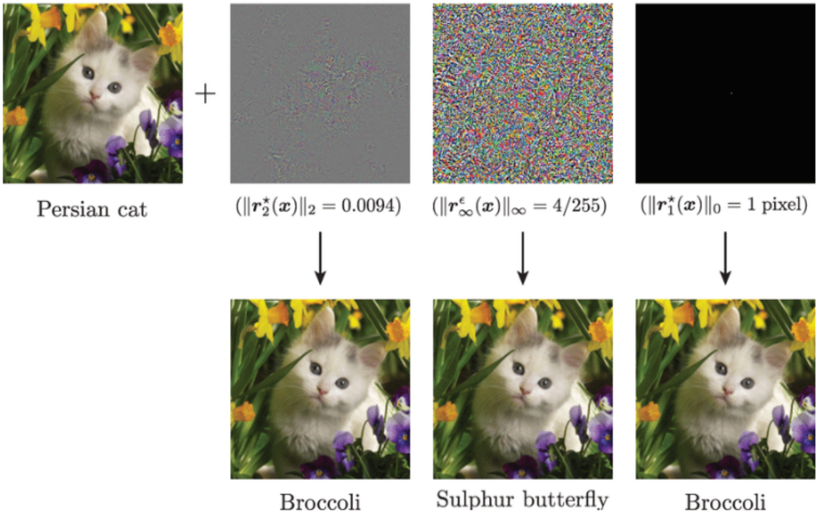

Persian cat
+
$(\|r_2^\star(x)\|_2 = 0.0094)$
$(\|r_\infty^\epsilon(x)\|_\infty = 4/255)$
$(\|r_1^\star(x)\|_0 = 1 \text{ pixel})$
Broccoli
Sulphur butterfly
Broccoli

Figure: Example of adversial atatck

Robustness results

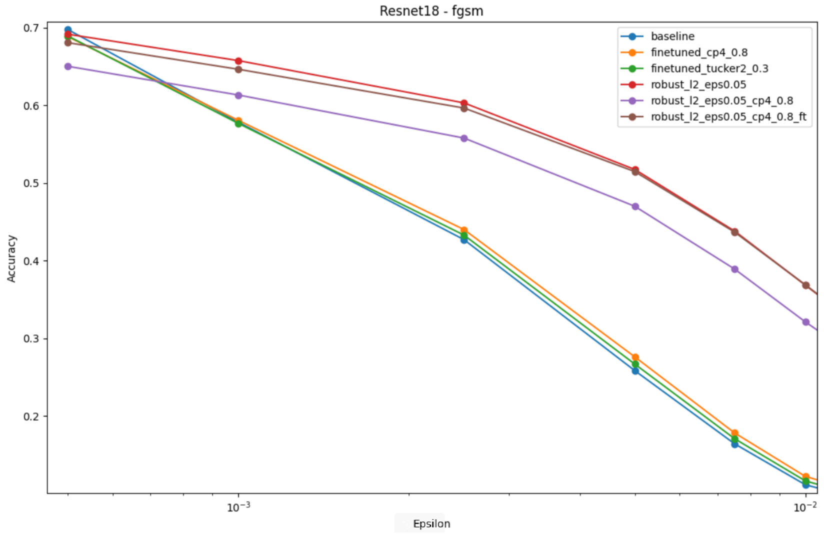


Figure: Robustness results

Transfer learning

- Neural networks have large numbers of parameters and therefore require a lot of data to be trained.
- Finding a large amount of data of good quality for a specific task can be difficult and expensive. Moreover, training a neural network from scratch can be expensive in terms of computation.
- Transfer learning is a technique that allows to use the knowledge acquired by a neural network on a task to perform another task.
- The idea is to use the weights of a neural network trained on a task as a starting point for the training of a neural network on another task.

Transfer learning results

Therefore I chose the FVGC-Aircraft dataset as a first benchmark. This dataset is made of picture of differnt type of plane and each type of plane has its own class.

Model	Compression rate	Fine tuned	Accuracy Aircraft	Accuracy Imagenet
Googlenet			55.5	69.8
Googlenet	0.4	N	58.1	63.4
Googlenet	0.4	Y	57.7	70.0
Googlenet	0.8	N	58.2	23.6
Googlenet	0.8	Y	**58.3**	67.7
Resnet-18			56.4	69.8
Resnet-18	0.3	N	**60.9**	61.9
Resnet-18	0.3	Y	60.5	68.9
Resnet-18	0.6	N	60.5	18.8
Resnet-18	0.6	Y	57.9	66.3

Table: Results of transfer learning from Imagenet to FVGC-Aircraft dataset.

Conclusion

- We used RTT as a framework to systematically combine different algorithms of signal detection.
- Furthermore, we have tested a meta-learning approach to our problem of finding optimal combination of signal detection algorithms. This approach showed an improvement over state-of-the-art algorithms.
- One of the possible compression techniques of deep neural networks is tensor decomposition.
- We proposed an adaptation of the popular tensor decomposition algorithm ALS that efficiently takes into account prior probability distribution of the data and that shows promising results.
- We provided experimental evidence that the robustness of the compressed networks is similar to the original network and that compressed networks can be used as a better starting point for transfer learning.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. (2014).
Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation.
In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Nakajima, S., Sugiyama, M., and Tomioka, R. (2010).
Global analytic solution for variational Bayesian matrix factorization.
Advances in Neural Information Processing Systems, 23.