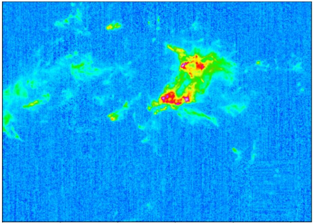
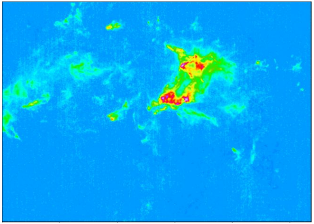
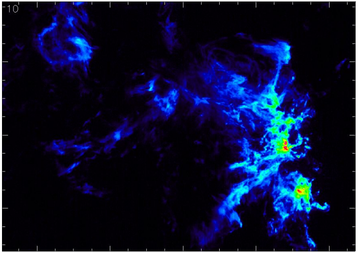
Efficient deep learning approaches to denoise radioastronomy line cubes and emulate astrophysical models

L. Einig, J. Pety, P. Palud, J. Chanussot, A. Roueff, M. Gerin
and the ORION-B consortium





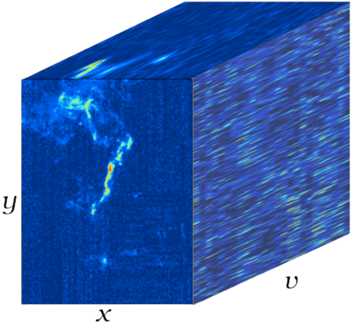
The ORION-B dataset



- Acquired by the wide-band receiver at the IRAM-30 m
- $\sim$ 1 000 h of observations
- Observations interpretation needs proper **images/data processing** and efficient **inference procedures**.

The ORION-B dataset

- $\sim$ 30 molecular line cubes for $J = 1$ - 0 transition
- Spatially and spectrally resolved
- 1074 $\times$ 758 profiles
- 240 velocity channels per cube



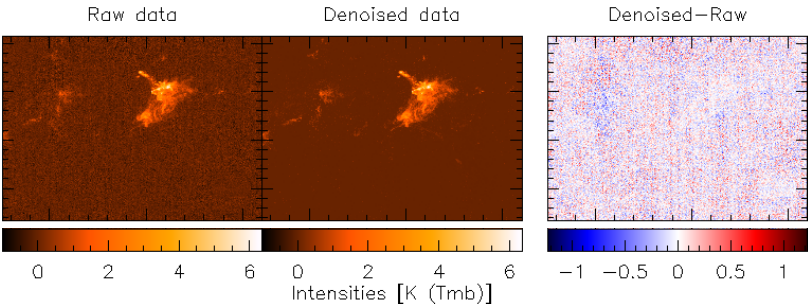
$^{13}\mathrm{CO}$ (1-0) line cube.

Denoising of radio astronomy line cubes

About denoising

Interest of denoising

- Increasing the signal-to-noise ratio is an important step to lead to discoveries.
- Necessary to find statistical relations between certain lines and physical parameters (otherwise hidden by noise).



Fully connected autoencoder-based denoising

Efficient for Earth remote sensing cubes (Licciardi+2015, Licciardi+2018).

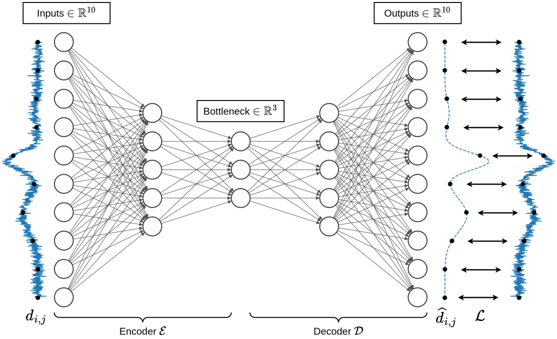


Figure: Example of an autoencoder neural network with extrinsic and intrinsic dimensions of 10 and 3, respectively.

A locally connected AE with prior information

- Noise is pixel dependent, spectrally and spatially correlated
  $\longrightarrow$ false signal
- Unlike Earth remote sensing cubes, very low information redundancy
  $\longrightarrow$ need to make the best use of it

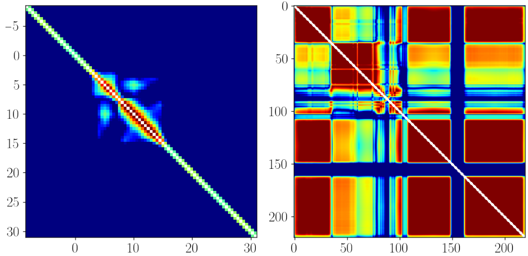


Figure: $^{13}\mathrm{CO}$ line and Earth image “Indian Pines” correlation matrices

A locally connected AE with prior information

Proposition 1: Loss function to help the network

- The redundancy is unsufficient for proper denoising.
- We give more information to the network with a loss function depending on a **prior**.
- This prior is taken from a **3D segmentation method**.

$$\mathcal{L}(\widehat{d}_{i,j}, d_{i,j}) = \frac{1}{K}\sum_{k=1}^{K}\begin{cases} \frac{(\widehat{d}_{i,j,k} - d_{i,j,k})^2}{\sigma_{i,j}} & \text{if probably signal + noise} \\ \left|\frac{\widehat{d}_{i,j,k}}{\sigma_{i,j}}\right|^q & \text{if probably only noise} \end{cases}$$

with $q \in]0,1]$ an hyperparameter than controls the sparcity.

A locally connected AE with prior information

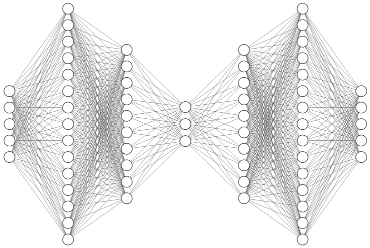
- Distant channels share almost no information
  $\longrightarrow$ most of the weights are useless, or even counter-productive

A locally connected AE with prior information

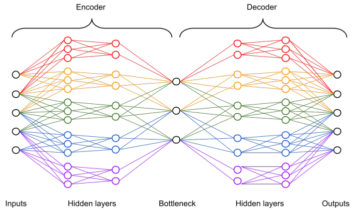
- Distant channels share almost no information
  $\longrightarrow$ most of the weights are useless, or even counter-productive

Proposition 2: Locally connected architecture

We propose this kind of architecture where distant channels cannot be combined together.



Example of fully connected AE

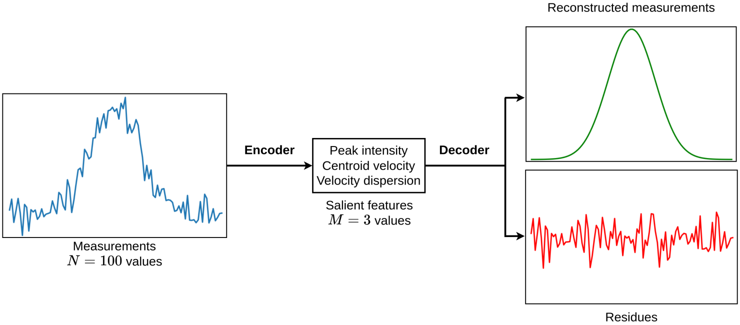


Example of locally connected AE

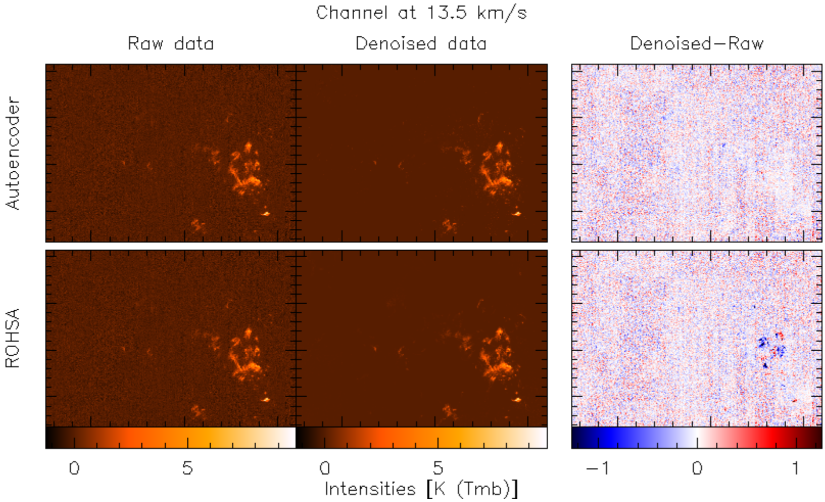
Gaussian fitting method ROHSA

Comparison with state-of-the-art ROHSA gaussian fitting method (Marchal+2019):

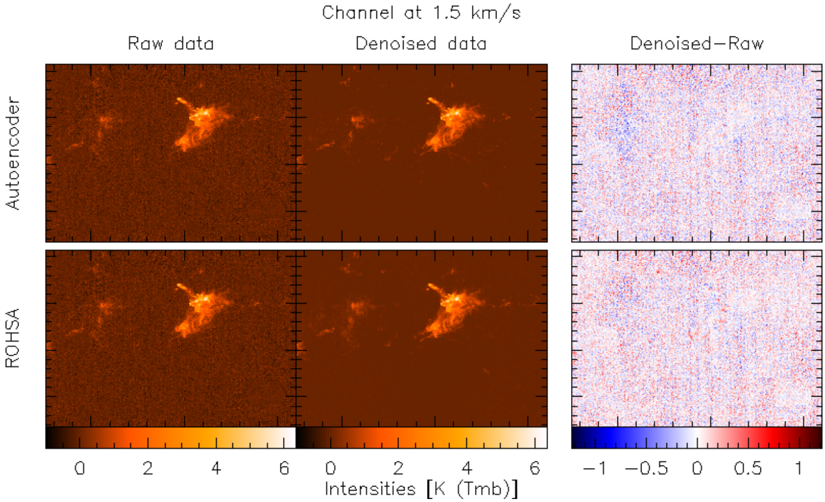
- Spatially constrained **Gaussian decomposition** of profiles.
- Reconstruction/decomposition can perform **denoising**.



Denoising performances: residuals



Denoising performances: residuals



Emulation of astrophysical codes

Interstellar medium and numerical simulations

Numerical simulations are widely used in order to model the ISM and compare theory with observations.

Interstellar medium and numerical simulations

Numerical simulations are widely used in order to model the ISM and compare theory with observations.

Computation time is often prohibitive for inference procedures.

Interstellar medium and numerical simulations

Numerical simulations are widely used in order to model the ISM and compare theory with observations.

Computation time is often prohibitive for inference procedures.

Usual solutions

Interstellar medium and numerical simulations

Numerical simulations are widely used in order to model the ISM and compare theory with observations.

Computation time is often prohibitive for inference procedures.

Usual solutions

- Interpolation methods:
  1. Nearest point in grid (Sheffer+2011; Sheffer+2013; Joblin+2018)
  2. SciPy interpolation (Wu+2018; Ramambason+2022)

Interstellar medium and numerical simulations

Numerical simulations are widely used in order to model the ISM and compare theory with observations.

Computation time is often prohibitive for inference procedures.

Usual solutions

- Interpolation methods:
  1. Nearest point in grid (Sheffer+2011; Sheffer+2013; Joblin+2018)
  2. SciPy interpolation (Wu+2018; Ramambason+2022)
- Regression-based approximations:
  1. *k*-nearest neighbors (Smirnov-Pinchukov+2022)
  2. Random forests (Bron+2021)
  3. Neural networks (de Mijolla+2019; Holdship+2021; Grassi+2022)

Interstellar medium and numerical simulations

Numerical simulations are widely used in order to model the ISM and compare theory with observations.

Computation time is often prohibitive for inference procedures.

Usual solutions

- Interpolation methods:
  1. Nearest point in grid (Sheffer+2011; Sheffer+2013; Joblin+2018)
  2. SciPy interpolation (Wu+2018; Ramambason+2022)
- Regression-based approximations:
  1. *k*-nearest neighbors (Smirnov-Pinchukov+2022)
  2. Random forests (Bron+2021)
  3. Neural networks (de Mijolla+2019; Holdship+2021; Grassi+2022)

  $\rightarrow$ Less complex, so faster and allow more training data

A Meudon PDR approximation as a template

The Meudon PDR code

- Emulates **photo-dissociation regions** (PDRs) at equilibrium.
- This version: **4** inputs $\longmapsto \sim$ **5 000** spectral lines intensities
- Execution time $\sim$ **6 hours** and may yield **anomalies**.
- Predictions **directly comparable with observations**.

A Meudon PDR approximation as a template

The Meudon PDR code

- Emulates **photo-dissociation regions** (PDRs) at equilibrium.
- This version: **4** inputs $\longmapsto \sim$ **5 000** spectral lines intensities
- Execution time $\sim$ **6 hours** and may yield **anomalies**.
- Predictions **directly comparable with observations**.

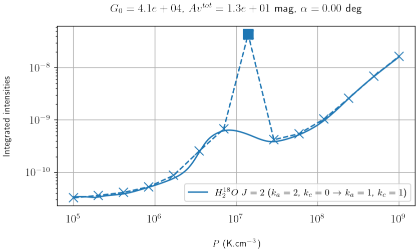
Overall: <u>representative example</u> of ISM numerical simulations.

A Meudon PDR approximation as a template

The Meudon PDR code

- Emulates **photo-dissociation regions** (PDRs) at equilibrium.
- This version: **4** inputs $\longmapsto \sim$ **5 000** spectral lines intensities
- Execution time $\sim$ **6 hours** and may yield **anomalies**.
- Predictions **directly comparable with observations**.

Overall: representative example of ISM numerical simulations.



Evolutions of a standard multilayer perceptron

Start: A multilayer perceptron to approach the simulations.

Evolutions of a standard multilayer perceptron

Start: A multilayer perceptron to approach the simulations.

Proposition 1: ignoring anomalies

Anomalies $\neq$ well-modeled points with sensitive behavior!

1. Training with a **robust loss** (e.g., Cauchy) to detect badly reconstructed points.
2. Use physics knowledge to determine anomalies among them.
3. New training from scratch with a **masked non-robust loss function** (e.g., MSE), **ignoring the abnormal outputs**.

Evolutions of a standard multilayer perceptron

Start: A multilayer perceptron to approach the simulations.

Proposition 1: ignoring anomalies

Anomalies $\neq$ well-modeled points with sensitive behavior!

1. Training with a **robust loss** (e.g., Cauchy) to detect badly reconstructed points.
2. Use physics knowledge to determine anomalies among them.
3. New training from scratch with a **masked non-robust loss function** (e.g., MSE), **ignoring the abnormal outputs**.

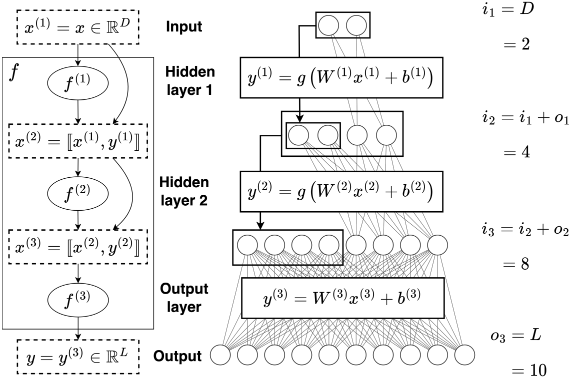
Proposition 2: outputs clustering to divide and conquer

Since outputs can be grouped into clusters by their similarity, it's efficient to train **dedicated networks** side by side for each cluster.

Evolutions of a standard multilayer perceptron

Proposition 3: reuse intermediate computations

As some outputs can be computed from other outputs, keeping track of **intermediate results** optimizes network capacities.



Results on the Meudon PDR code

		Method	Error factor mean	99% per.	max	Memory (MB)	Speed (ms)
No outlier removal		near. neighbor	×13.1	×11.3	×3e5	1 650	62
		linear	15.7	×2.3	×143	1 650	1.5e3
	spline	linear	15.7	×2.3	×144	1 650	...
		cubic	11.2	×2.2	×122	1 650	...
		quintic	19.1	×2.9	×304	1 650	...
	RBF	linear	10.2	96.8	×99	1 650	1.1e4
		cubic	10.4	×2.1	×112	1 650	1.1e4
		quintic	10.9	×2.1	×118	1 650	1.1e4
	ANN	R	7.3	64.8	×**81**	**118**	**12**
		R+P	**6.2**	**49.7**	×84	**118**	13

Error factor

→ Symmetrized relative error

$$\mathrm{err}_{\%} = 100 \cdot \max\left(\frac{\hat{y}}{y}, \frac{y}{\hat{y}}\right)$$

Speed

→ Computation of a batch of 1 000 entries on a laptop.

R: regression by an ANN

P: polynomial expansion

C: lines clustering

D: dense architecture

Results on the Meudon PDR code

Method			Error factor mean	99% per.	max	Memory (MB)	Speed (ms)
No outlier removal		near. neighbor	×13.1	×11.3	×3e5	1 650	62
		linear	15.7	×2.3	×143	1 650	1.5e3
	spline	linear	15.7	×2.3	×144	1 650	…
		cubic	11.2	×2.2	×122	1 650	…
		quintic	19.1	×2.9	×304	1 650	…
	RBF	linear	10.2	96.8	×99	1 650	1.1e4
		cubic	10.4	×2.1	×112	1 650	1.1e4
		quintic	10.9	×2.1	×118	1 650	1.1e4
	ANN	R	7.3	64.8	×**81**	**118**	**12**
		R+P	**6.2**	**49.7**	×84	**118**	13

Error factor

→ Symmetrized relative error

$$\mathrm{err}_{\%} = 100 \cdot \max\left(\frac{\hat{y}}{y}, \frac{y}{\hat{y}}\right)$$

Speed

→ Computation of a batch of 1 000 entries on a laptop.

R: regression by an ANN

P: polynomial expansion

C: lines clustering

D: dense architecture

Results on the Meudon PDR code

		Method	Error factor mean	Error factor 99% per.	Error factor max	Memory (MB)	Speed (ms)
No outlier removal		near. neighbor	×13.1	×11.3	×3e5	1 650	62
		linear	15.7	×2.3	×143	1 650	1.5e3
	spline	linear	15.7	×2.3	×144	1 650	...
		cubic	11.2	×2.2	×122	1 650	...
		quintic	19.1	×2.9	×304	1 650	...
	RBF	linear	10.2	96.8	×99	1 650	1.1e4
		cubic	10.4	×2.1	×112	1 650	1.1e4
		quintic	10.9	×2.1	×118	1 650	1.1e4
	ANN	R	7.3	64.8	×**81**	**118**	**12**
		R+P	**6.2**	**49.7**	×84	**118**	13
Outlier removal on training set		near. neighbor	×13.1	×11.6	×3e5	1 650	62
		linear	15.9	×2.4	×143	1 650	1.5e3
	spline	linear	15.9	×2.4	×144	1 650	...
		cubic	11.1	×2.2	×120	1 650	...
		quintic	20.0	×2.7	×285	1 650	...
	RBF	linear	10.3	97.3	×97.5	1 650	1.1e4
		cubic	10.5	×2.0	×106	1 650	1.1e4
		quintic	10.9	×2.0	×114	1 650	1.1e4
	ANN	R	5.1	42.0	×**32.8**	118	12
		R+P	5.5	42.3	×41	118	13
		R+P+C	4.9	44.5	×44	51	14
		R+P+D	**4.5**	**33.1**	×33.8	125	**11**
		R+P+C+D	4.8	37.9	×37.6	**43**	14

Error factor

→ Symmetrized relative error

$$\mathrm{err}_{\%} = 100 \cdot \max\left(\frac{\hat{y}}{y}, \frac{y}{\hat{y}}\right)$$

Speed

→ Computation of a batch of 1 000 entries on a laptop.

R: regression by an ANN

P: polynomial expansion

C: lines clustering

D: dense architecture

Conclusions

Take-home messages

- AI benefits from rigorous data analysis and physical knowledge, for example here for denoising.
- Deep learning is efficient to emulate complex simulations, especially with additional constraints.

Take-home messages

- AI benefits from rigorous data analysis and physical knowledge, for example here for denoising.
- Deep learning is efficient to emulate complex simulations, especially with additional constraints.

- *Deep learning denoising by dimension reduction: Application to the ORION-B line cubes*, L. Einig, J. Pety et al, A&A, 2023.
- *Neural network-based emulation of interstellar medium models*, P. Palud, L. Einig et al, A&A, 2023.

- `https://github.com/einigl`
- `https://pypi.org/project/nnbma` $\longrightarrow$ `pip install nnbma`