Optimization meets machine learning for neuroimaging

Alexandre Gramfort

http://alexandre.gramfort.net

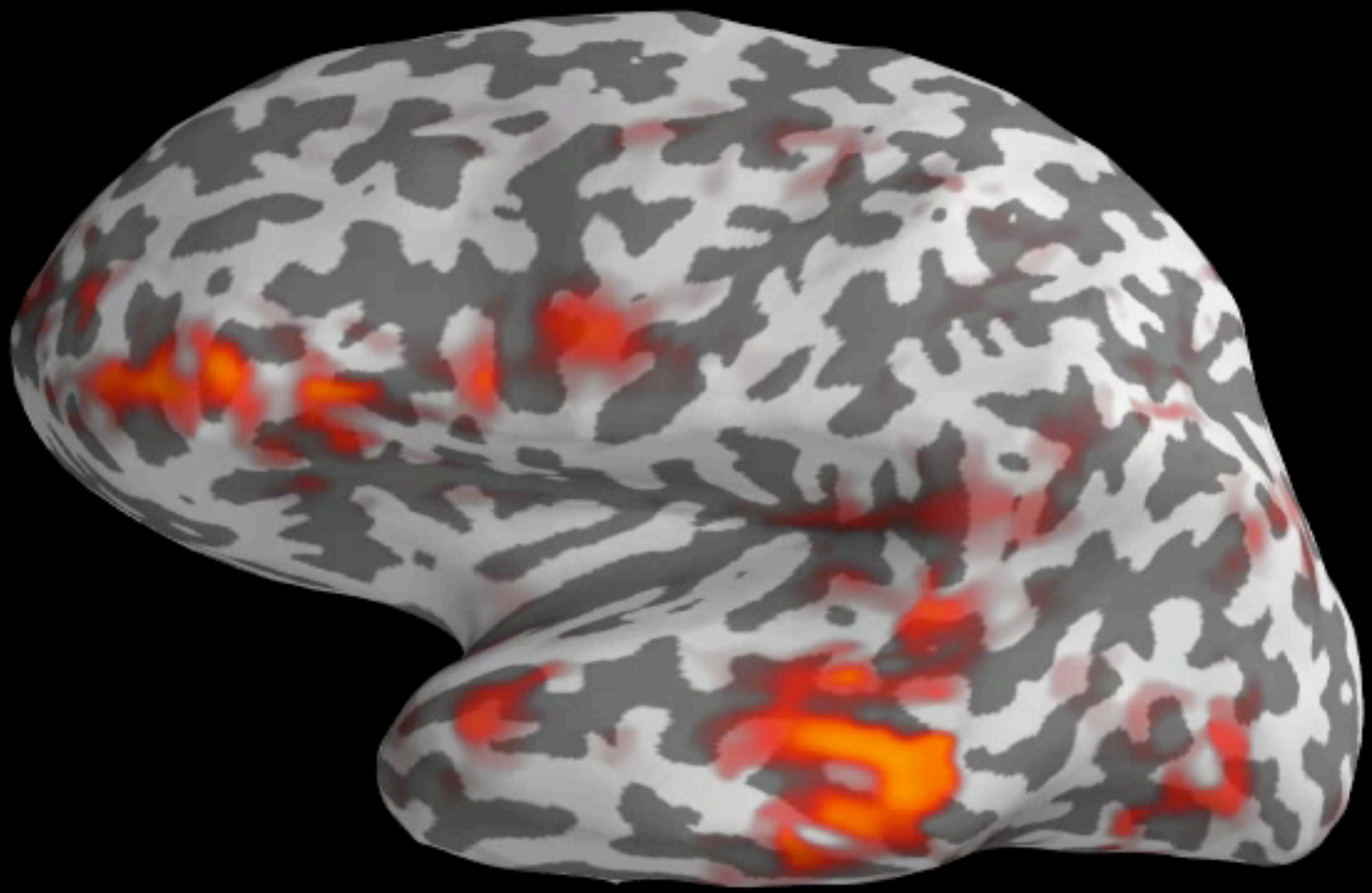
PARIETAL

informatics mathematics

université PARIS-SACLAY

Imaging and machine learning workshop - IHP 2019

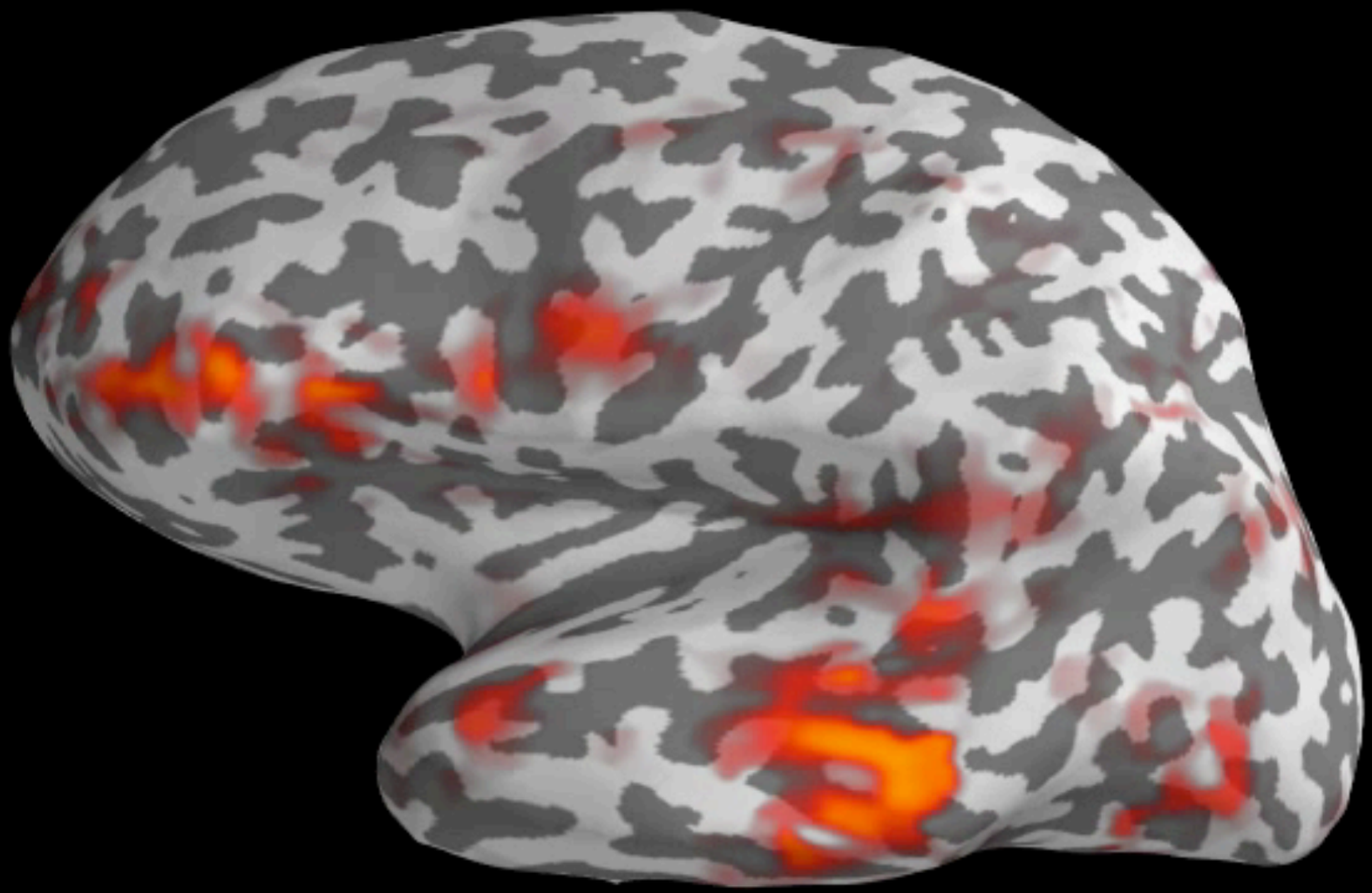
Source imaging with magnetoencephalography (MEG)



time=0.00 ms

http://youtu.be/Uxr5Pz7JPrs

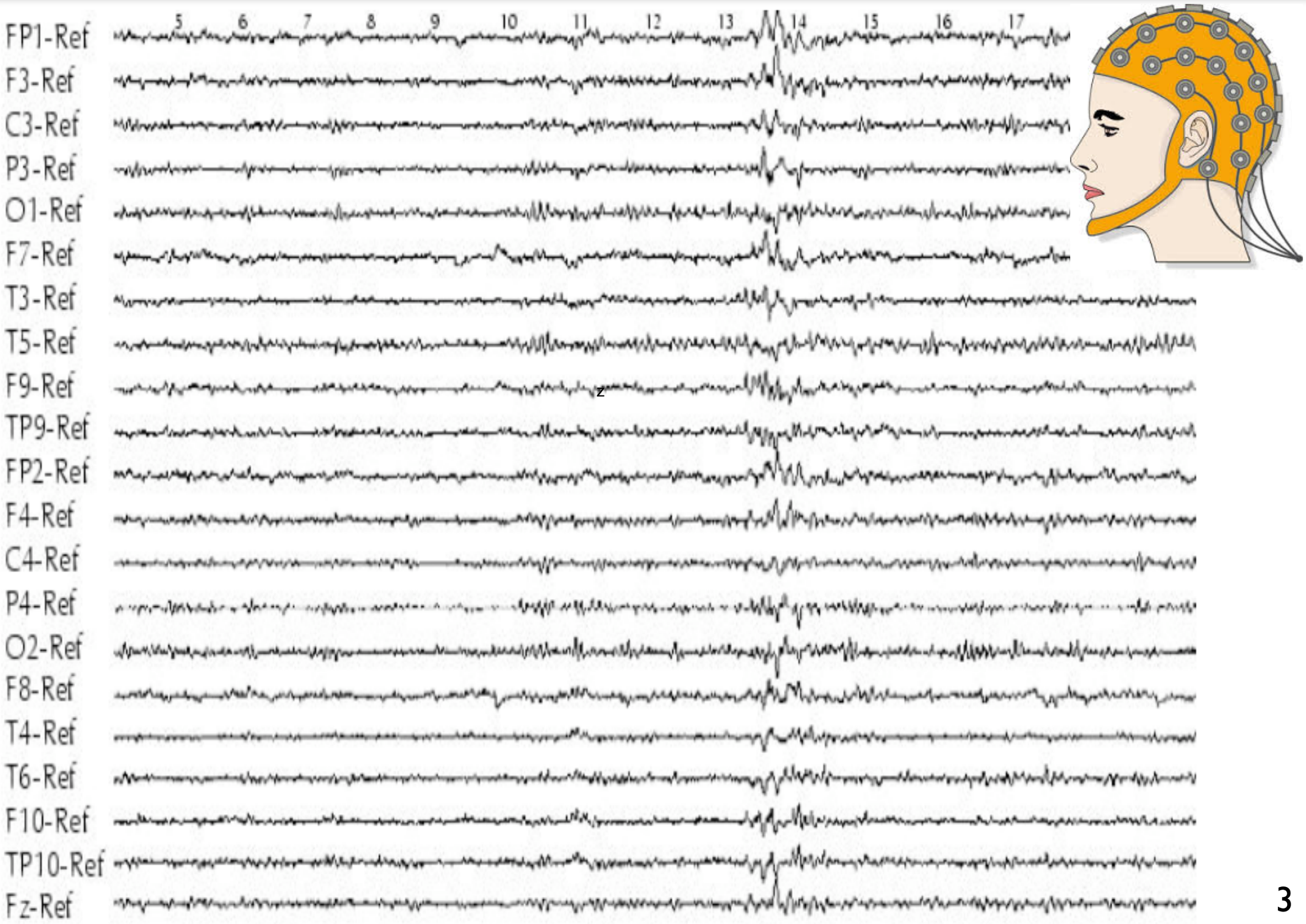
Source imaging with magnetoencephalography (MEG)



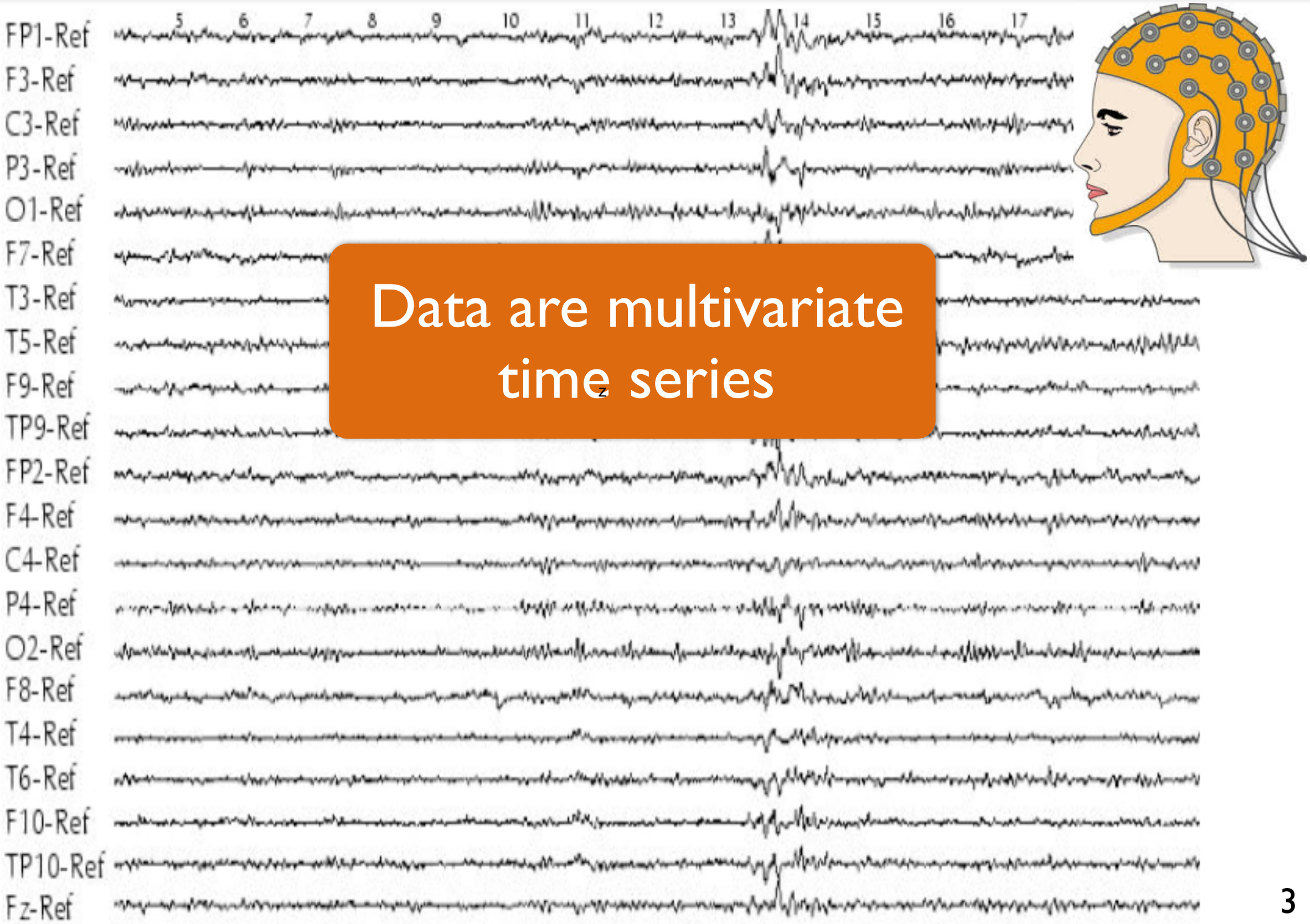
time=0.00 ms

http://youtu.be/Uxr5Pz7JPrs

Electroencephalography (EEG))

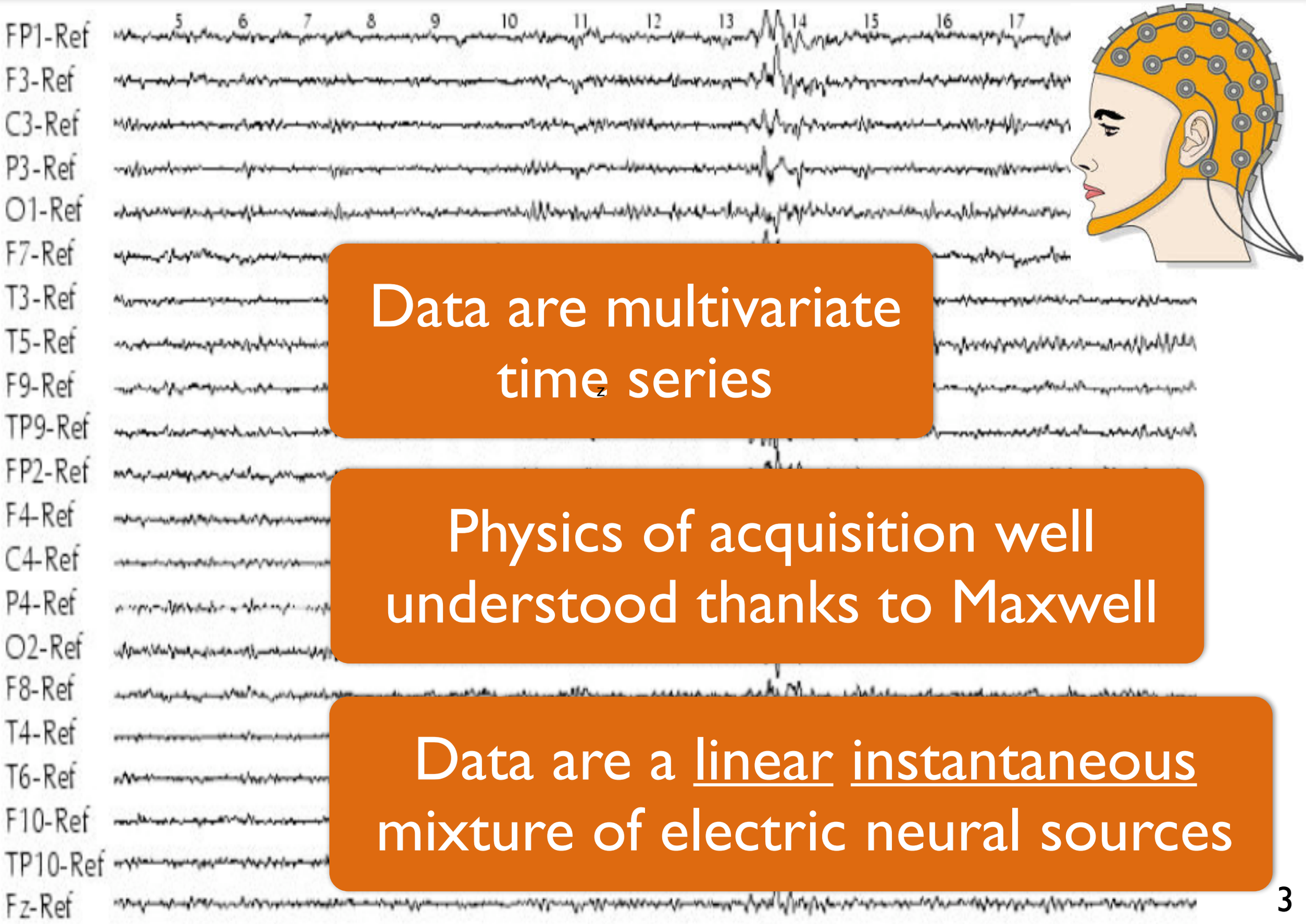


Electroencephalography (EEG))



Data are multivariate time series

Electroencephalography (EEG))

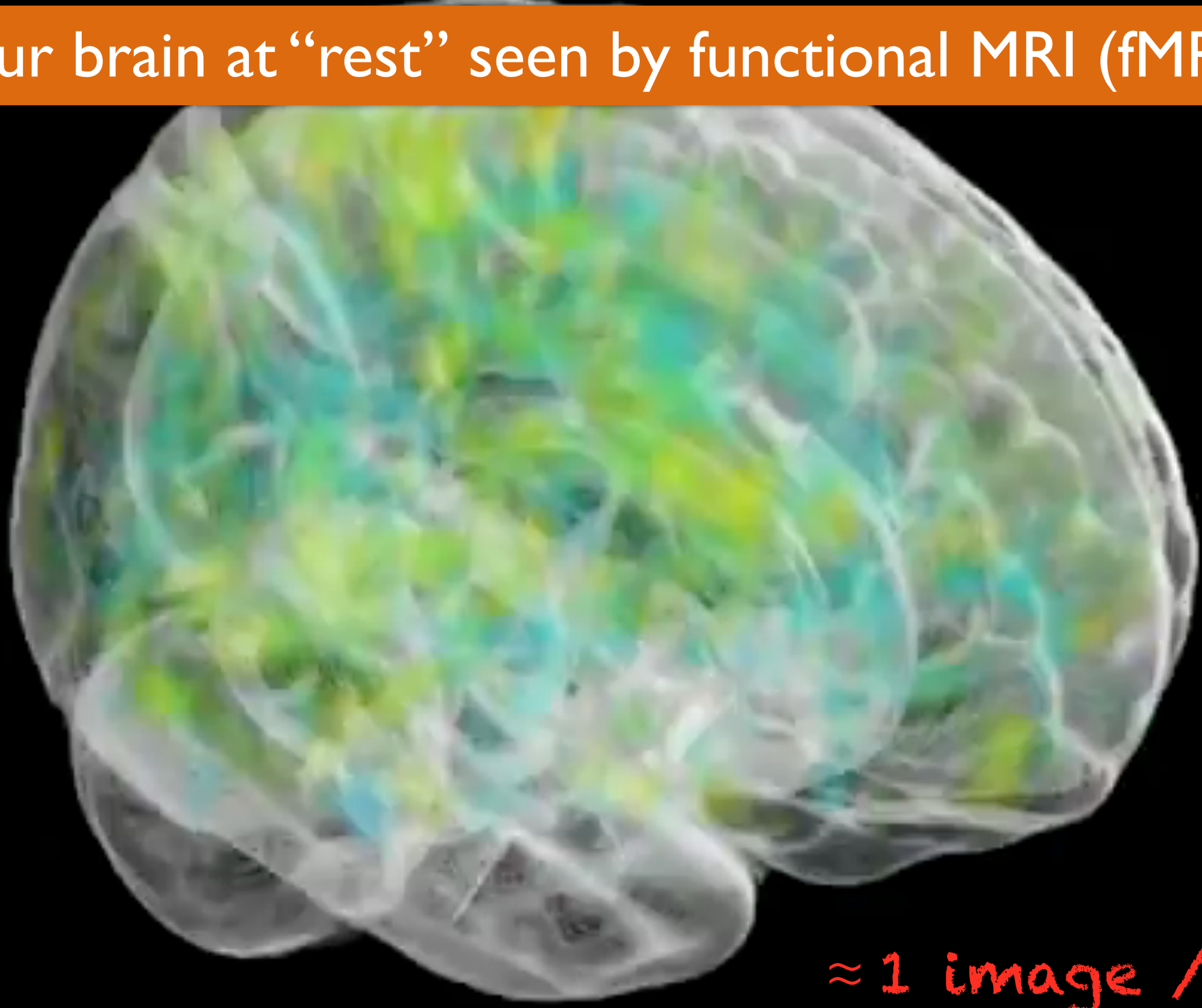


Data are multivariate time series

Physics of acquisition well understood thanks to Maxwell

Data are a linear instantaneous mixture of electric neural sources

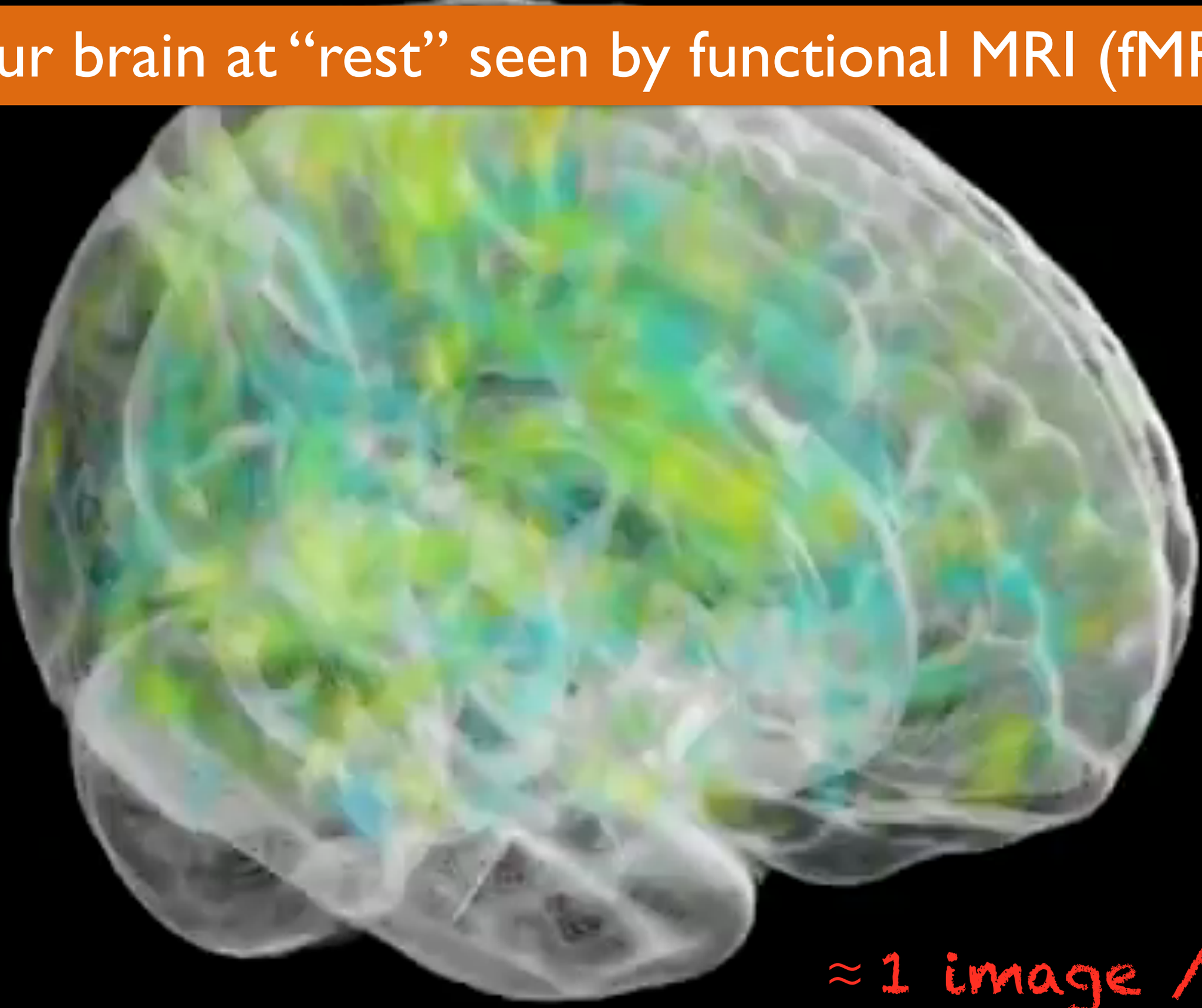
Your brain at "rest" seen by functional MRI (fMRI)



≈ 1 image / 2s

http://www.youtube.com/watch?v=uhCF-zlk0jY courtesy of Gael Varoquaux

Your brain at "rest" seen by functional MRI (fMRI)



≈ 1 image / 2s

http://www.youtube.com/watch?v=uhCF-zlk0jY courtesy of Gael Varoquaux

Question 1: What can unsupervised machine learning tell us about such data (the brain)?

Question 1: What can unsupervised machine learning tell us about such data (the brain)?

Question 2: Can we use the known physics to find adequate models?

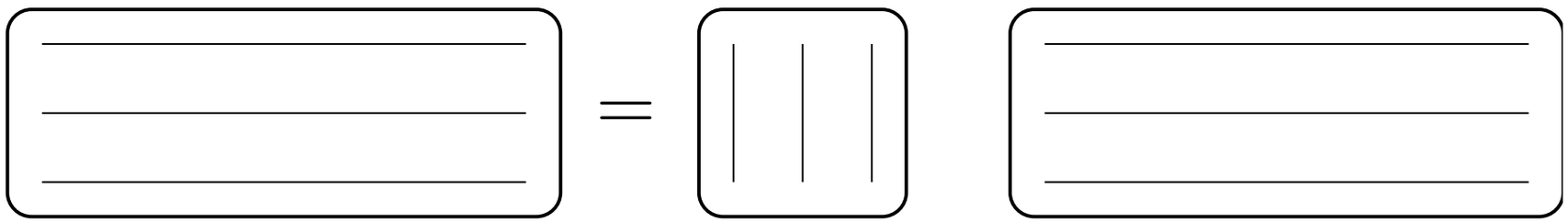
Question 1: What can unsupervised machine learning tell us about such data (the brain)?

Question 2: Can we use the known physics to find adequate models?

Question 3: Can we come up with efficient learning/optimization algorithms?

Linear ICA model

- **Assumption:** Observed signals are a linear mix of independent identically distributed signals.



$$X = A \quad S$$

- N: Number of signals
- T: Number of samples
- X: Observed signals. Size N x T
- S: Independent **sources** signals. Size N x T

Both A and S are unknown

[Jutten & Herault 91]

ICA vs Dictionary Learning

- **ICA:**

$$X = AS$$

Uses independence assumption

- **Dictionary learning:**

$$\min_{A,S} \|X - AS\|_F^2 + \lambda \|S\|_p \qquad p \leq 1$$

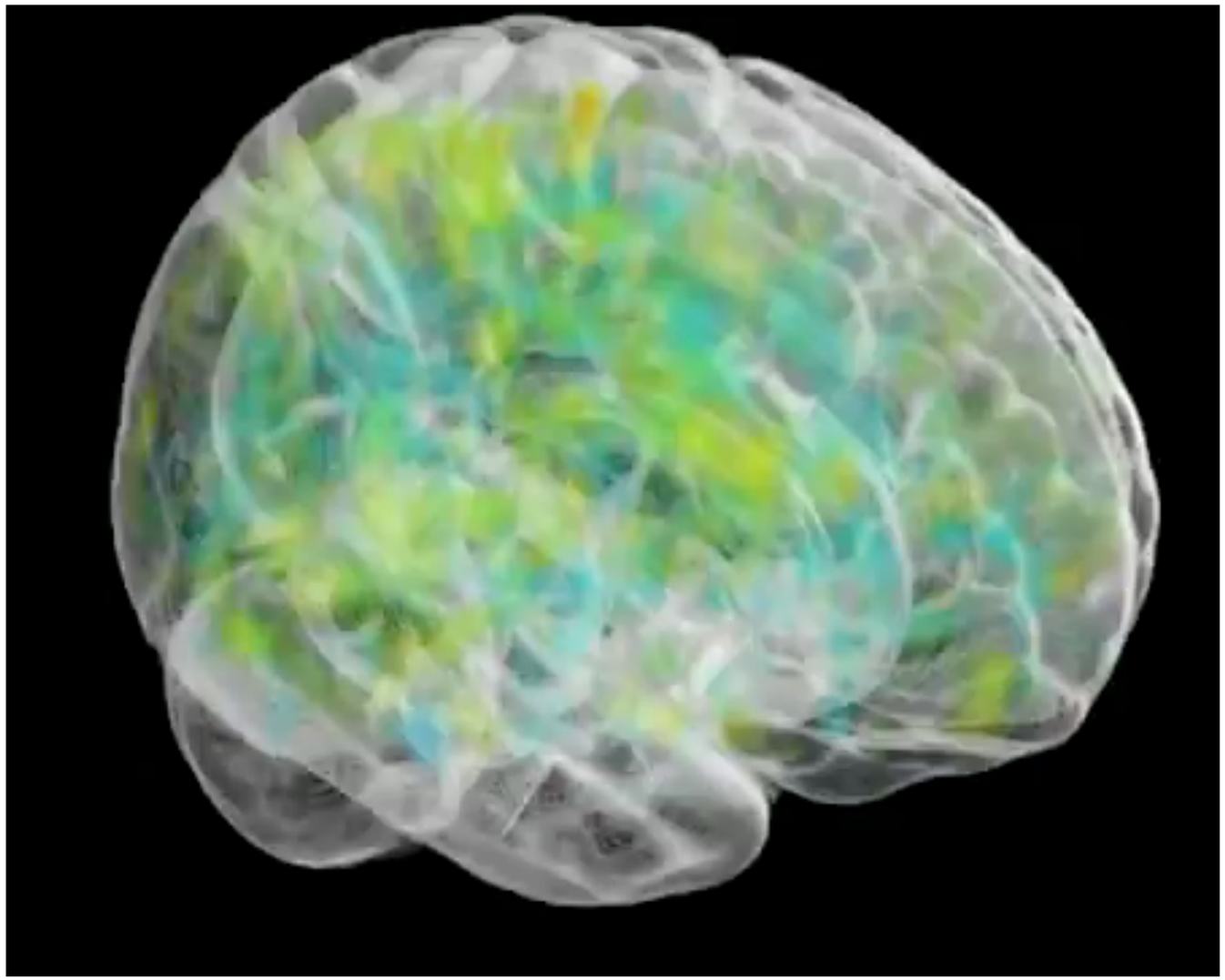
$$\text{s.t. } \|a_j\| \leq 1$$

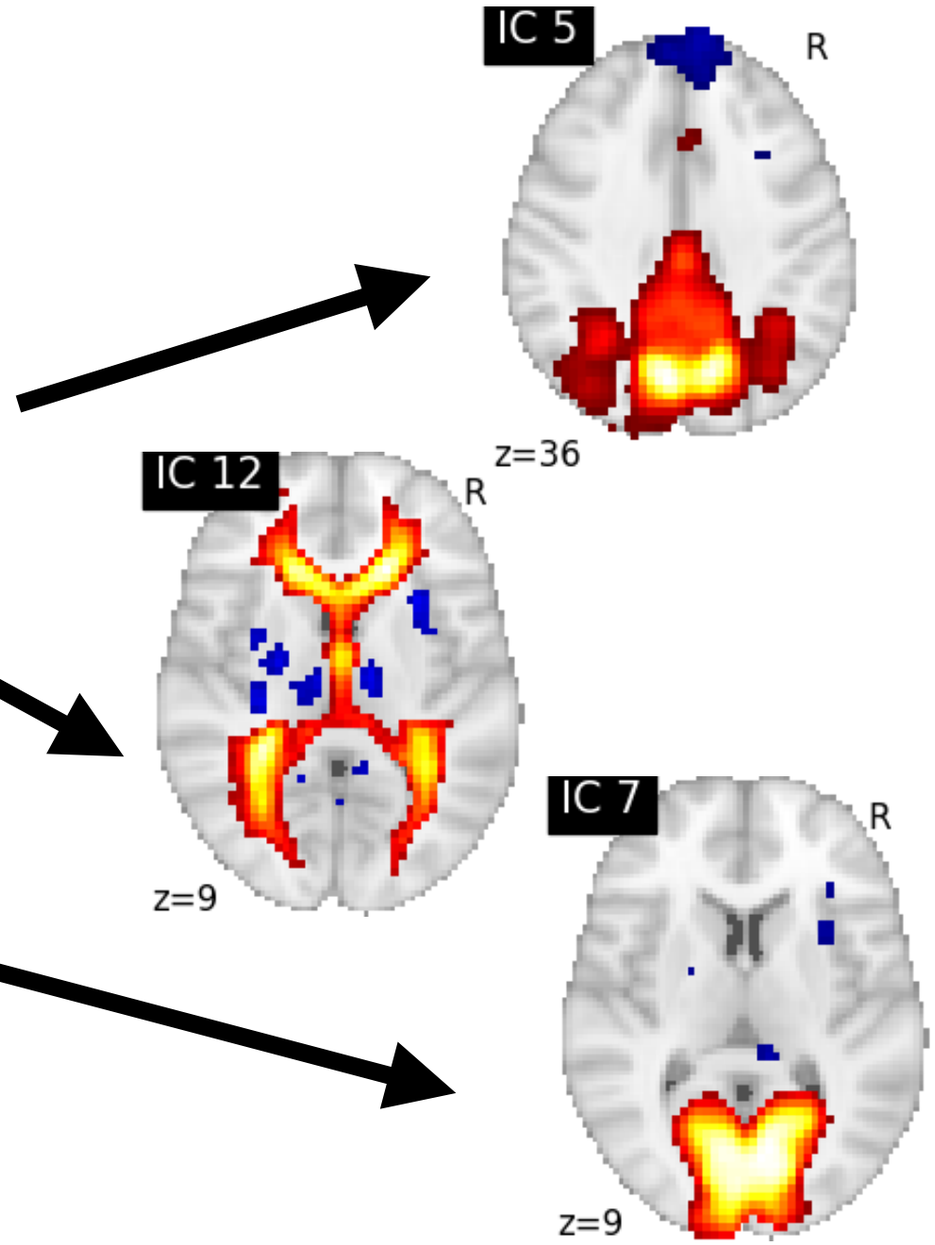
Uses sparsity assumption

[Olshausen et al., Elad et al., Mairal et al. etc…]

ICA on fMRI data

- Used in neuroimaging to find "brain networks" (not neural networks…) in functional MRI data

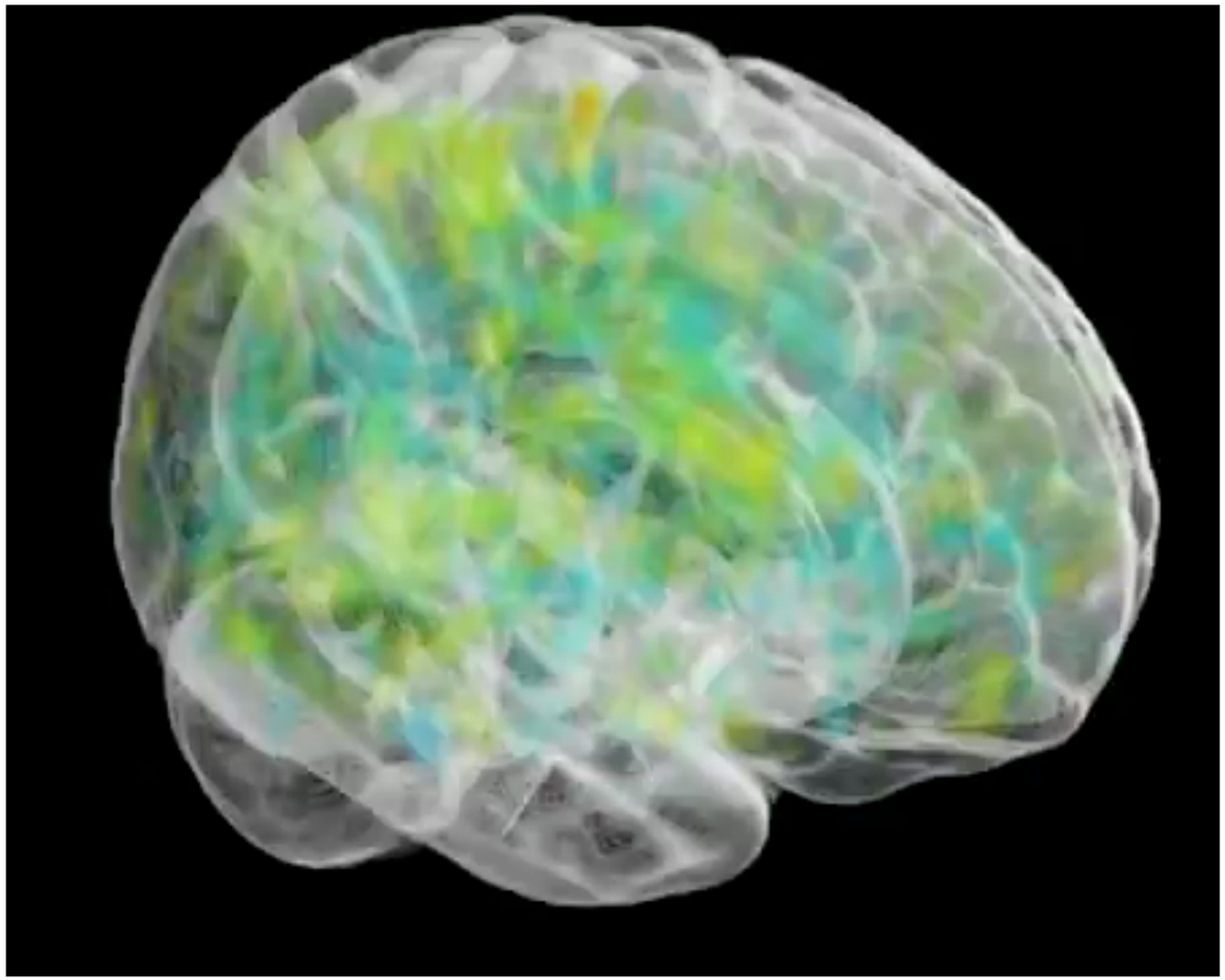


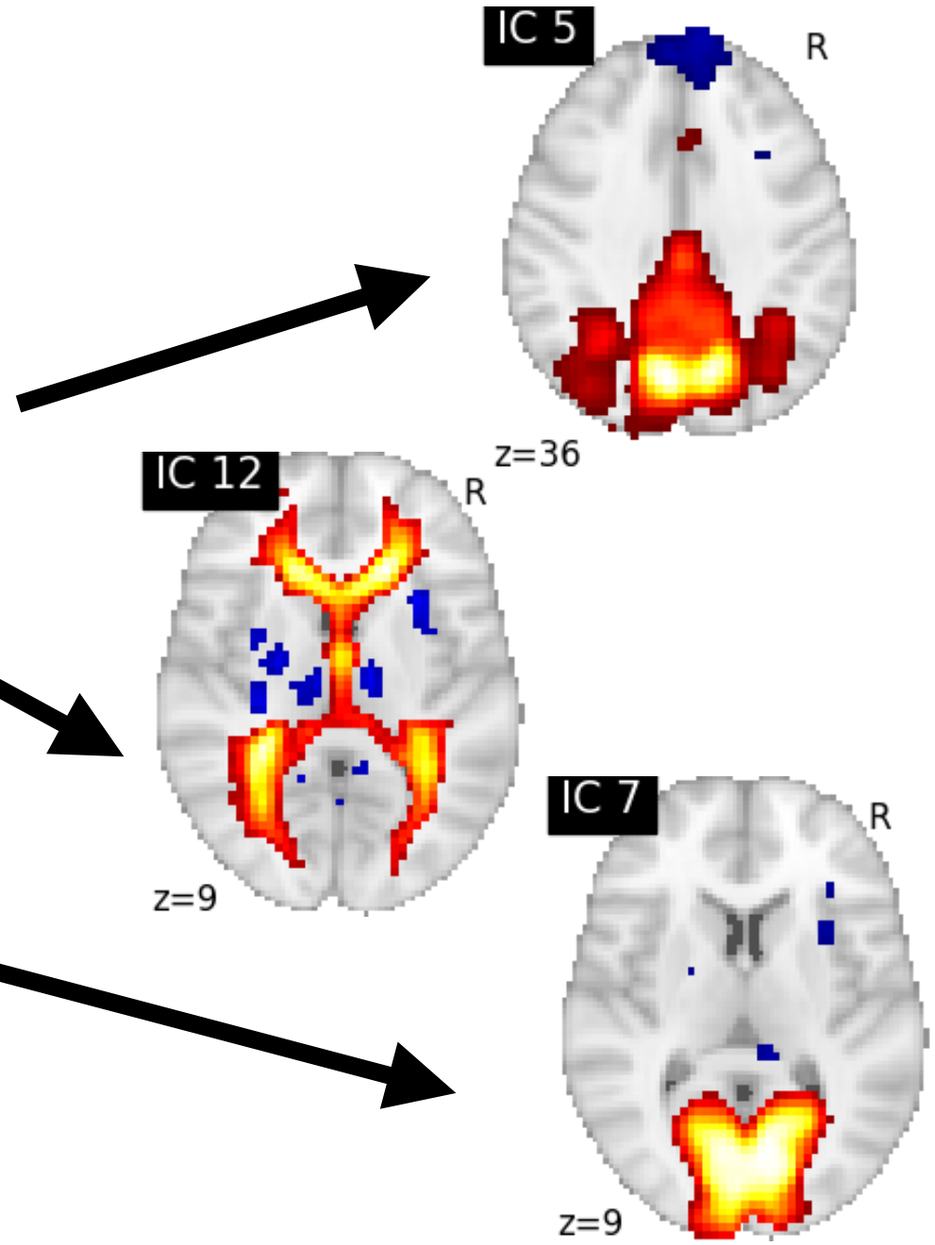


Source: http://nilearn.github.io/auto_examples/03_connectivity/plot_canica_resting_state.html

ICA on fMRI data

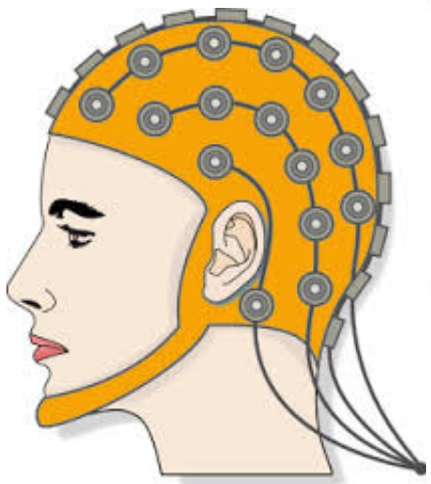
- Used in neuroimaging to find "brain networks" (not neural networks…) in functional MRI data

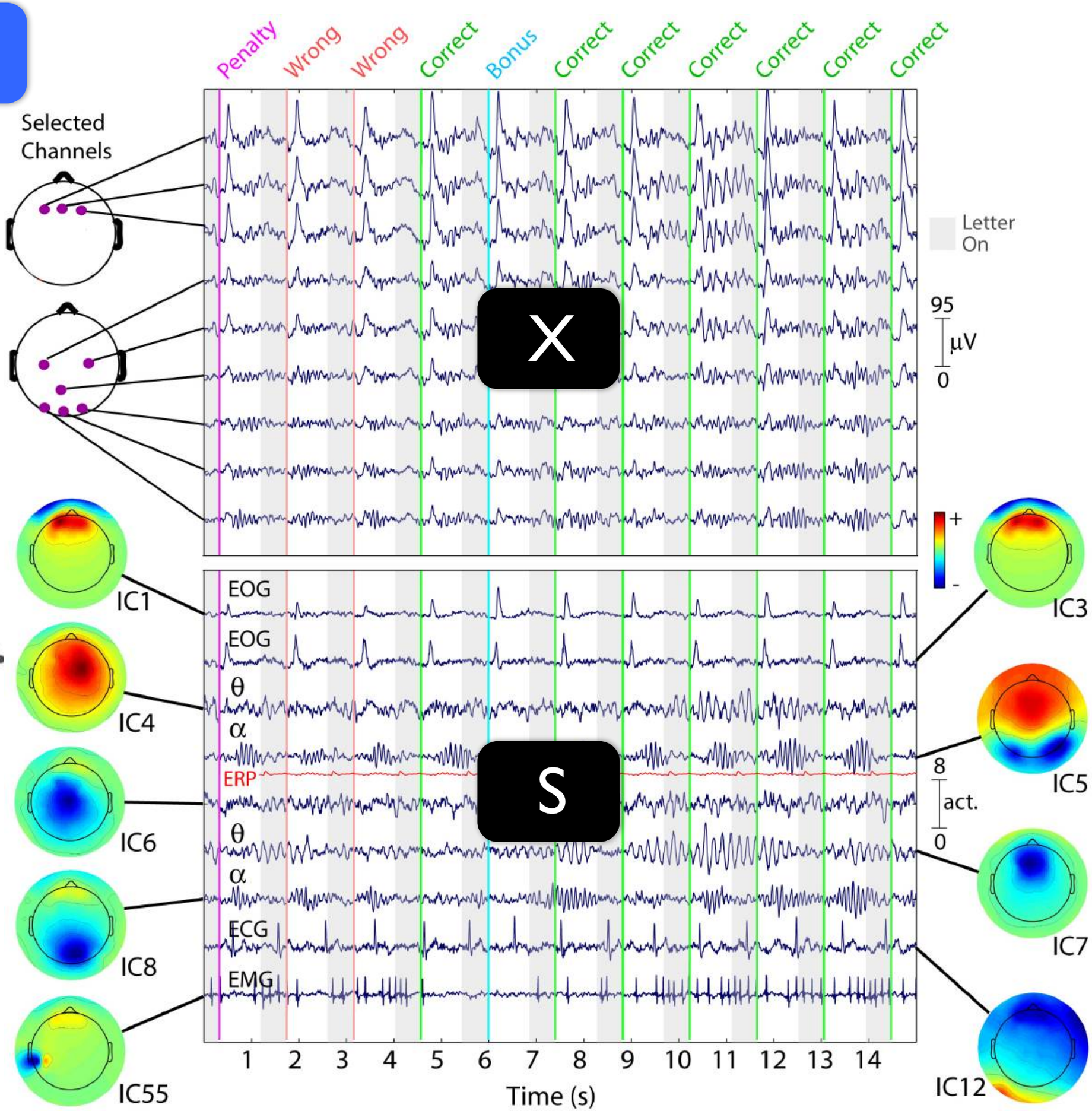




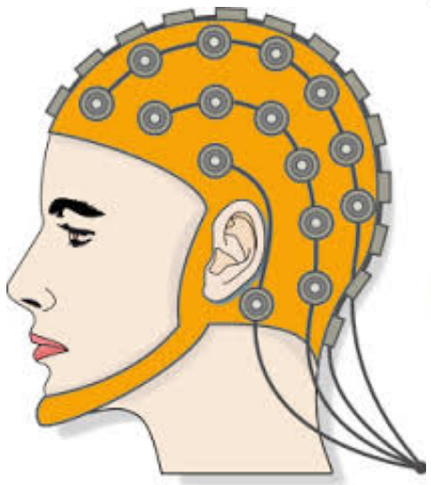
Source: http://nilearn.github.io/auto_examples/03_connectivity/plot_canica_resting_state.html

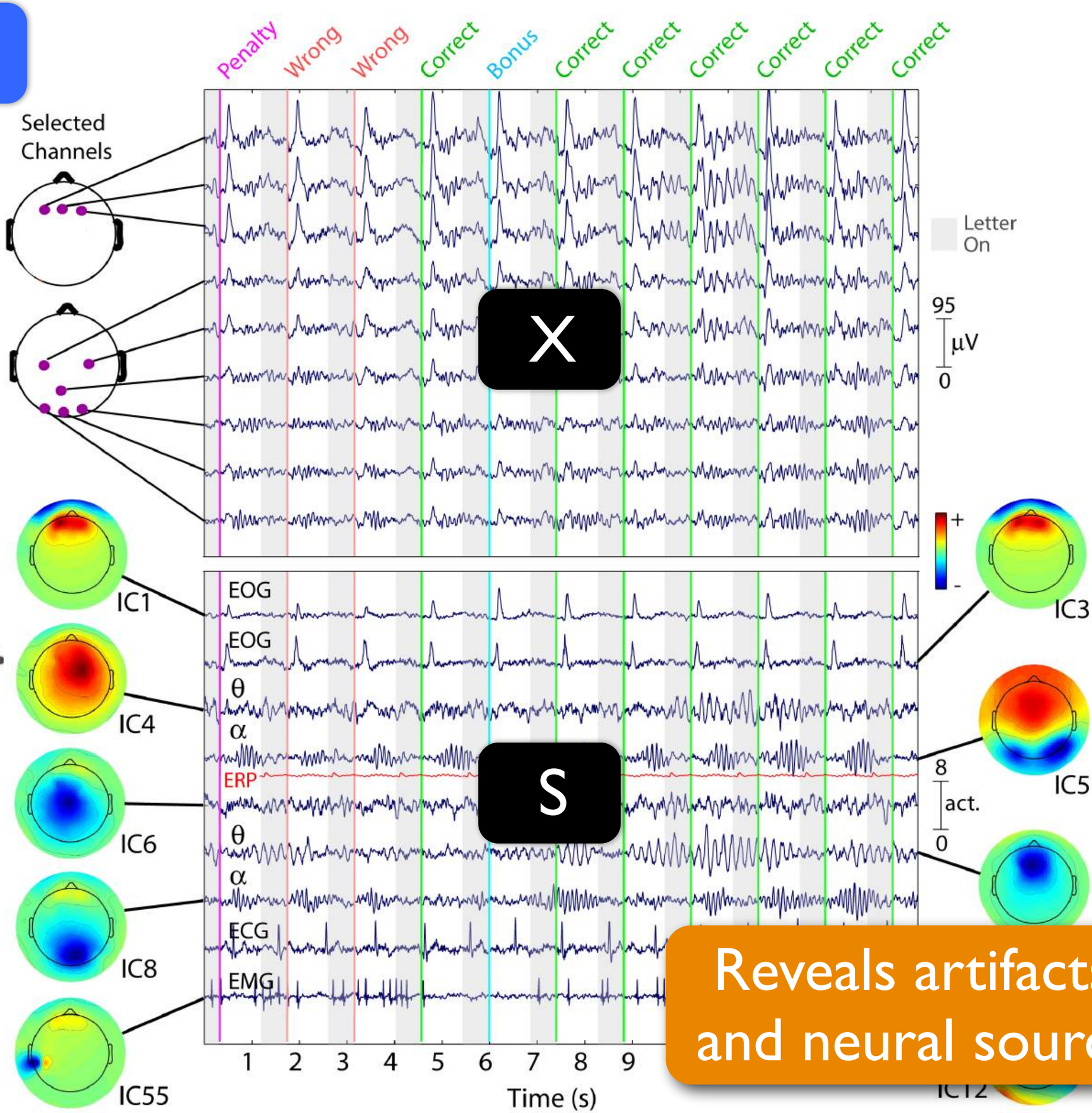
On EEG




Penalty
Wrong
Wrong
Correct
Bonus
Correct
Correct
Correct
Correct
Correct
Correct
Selected Channels
X
Letter On
95
μV
0
EOG
EOG
θ
α
ERP
S
θ
α
ECG
EMG
8
act.
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Time (s)
IC1
IC3
IC4
IC5
IC6
IC7
IC8
IC12
IC55

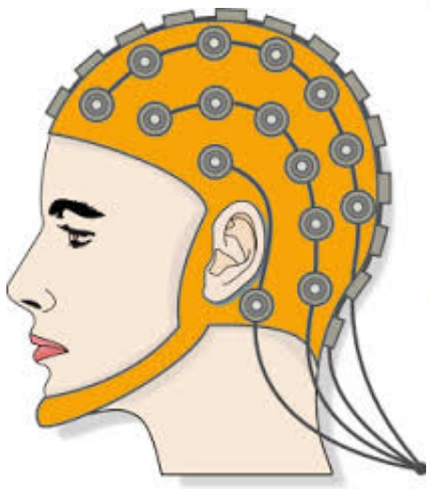
On EEG

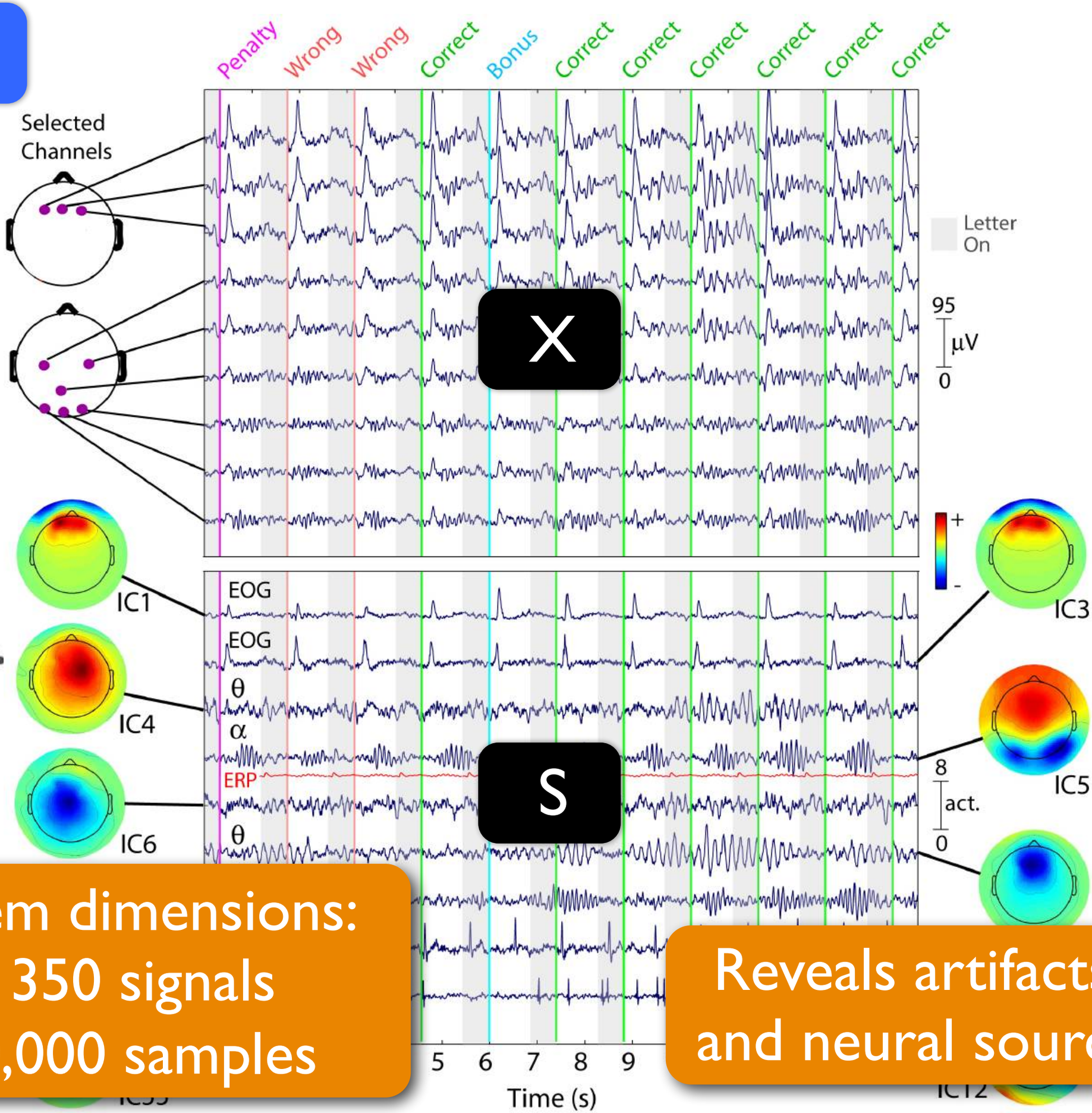




Reveals artifacts and neural sources

On EEG





Problem dimensions:

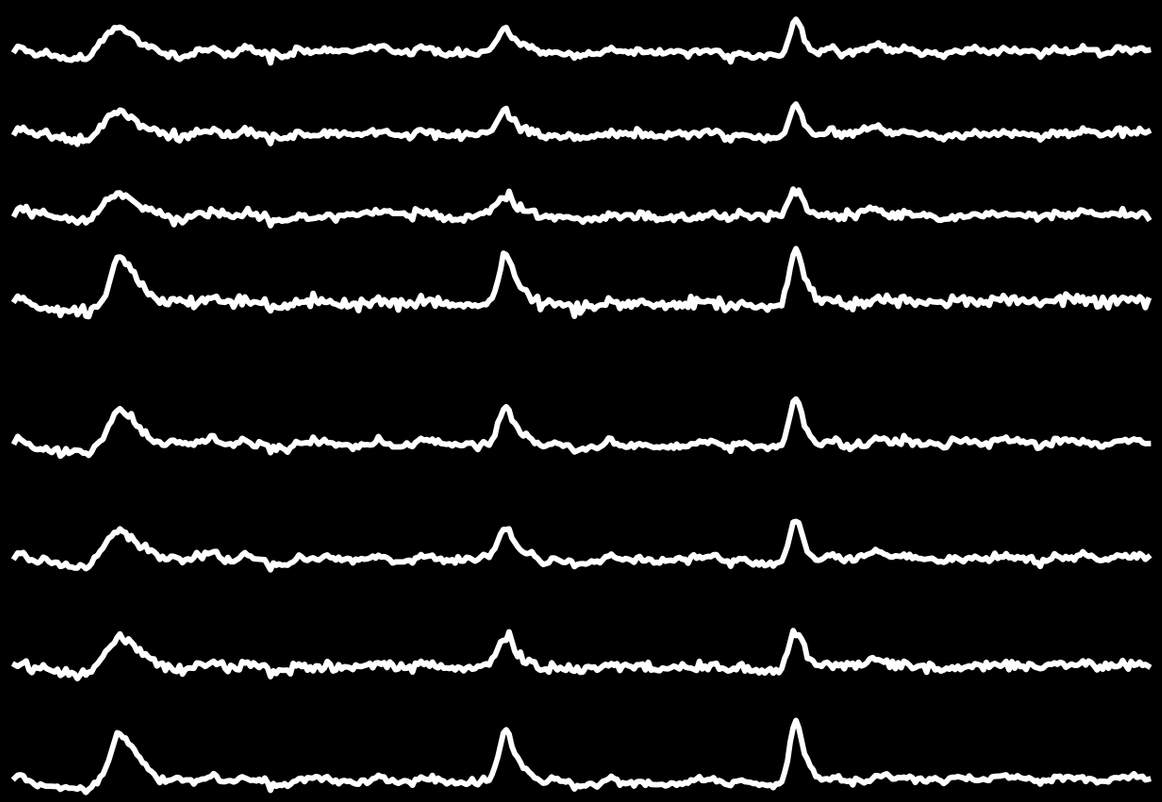
- 30 to 350 signals
- ≈100,000 samples

Reveals artifacts and neural sources

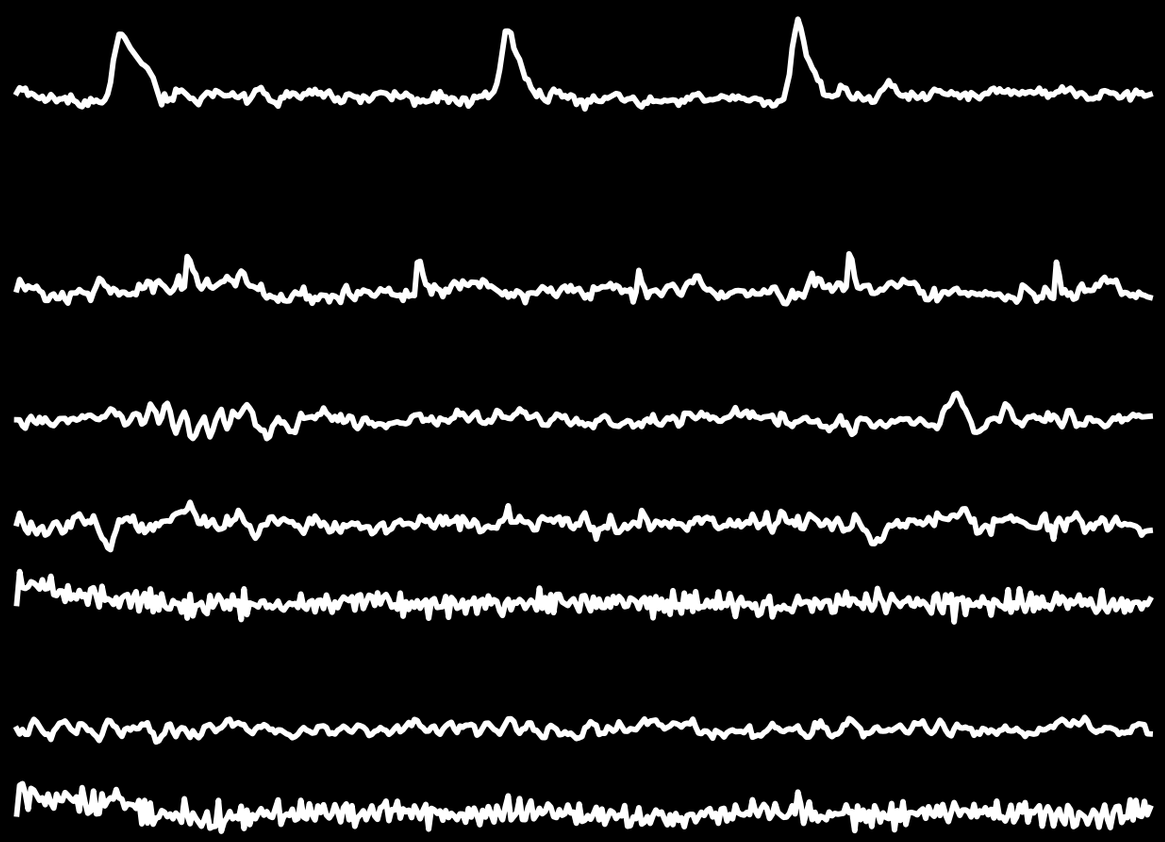
1

How to accelerate Independent Component Analysis (ICA) solvers?

Observations (raw EEG)



ICA recovered sources



[Faster independent component analysis by preconditioning with Hessian approximations, P. Ablin, J.-F. Cardoso & A. Gramfort 2017 IEEE Trans. Signal Processing]

[EM algorithms for ICA, P. Ablin, A. Gramfort, J.F. Cardoso and F. Bach, AISTATS 2019]

Code: https://pierreablin.github.io/picard

Statistical Principles of ICA

$$X = AS$$

Objective:

Find W s.t. $Y = WX$ has maximally independent rows.

Note:

If more than 2 Gaussian sources, ICA problem is not possible.

[Comon 94]

Remark:

Different ways to quantify independence lead to different methods.

Maximum likelihood & ICA

Maximum Likelihood Formulation

- **Model**: $Y = WX$ has independent rows

$$Y = [y_1, \ldots, y_N]^\top \qquad p_i \text{ p.d.f. of } y_i$$

- Independence:

$$p(Y(t)) = p_1(y_1(t))p_2(y_2(t)) \ldots p_N(y_N(t))$$

- Likelihood of a time sample of X:

$$p(X(t)) = |\det(W)|p_1(y_1(t))p_2(y_2(t)) \ldots p_N(y_N(t))$$

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \sum_{i=1}^{N} E_x[\log(p_i([Wx]_i))]$$

[Pham & Garat 1997]

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \sum_{i=1}^{N} E_x[\log(p_i([Wx]_i))]$$

[Pham & Garat 1997]

ICA boils down to minimizing an objective function

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \sum_{i=1}^{N} E_x[\log(p_i([Wx]_i))]$$

[Pham & Garat 1997]

ICA boils down to minimizing an objective function

- Infomax model *[Bell & Sejnowski 1995]*

$$\psi_i(\cdot) = -\log(p_i(\cdot))' = \tanh(\cdot/2)$$

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \sum_{i=1}^{N} E_x[\log(p_i([Wx]_i))]$$

[Pham & Garat 1997]

ICA boils down to minimizing an objective function

- Infomax model *[Bell & Sejnowski 1995]*

$$\psi_i(\cdot) = -\log(p_i(\cdot))' = \tanh(\cdot/2)$$

Most widely used ICA technique in neuroscience

Geometry of the problem

- non-convex (multiple minima)
- Optimization on the invertible matrices manifold
- Use of **relative changes** in solver iterations:

Absolute

$$W_{n+1} = W_n + dW$$

Relative

$$W_{n+1} = (I + \mathcal{E})W_n$$

n: iteration number

Relative gradient / Hessian

Relative matrix form Taylor expansion:

$$\mathcal{L}((I+\mathcal{E})W) = \mathcal{L}(W) + \langle G|\mathcal{E}\rangle + \frac{1}{2}\langle \mathcal{E}|H|\mathcal{E}\rangle + \mathcal{O}(||\mathcal{E}||^3)$$

- Gradient is a $N \times N$ matrix:

$$G_{ij} = E[\psi_i(y_i)y_j] - \delta_{ij}$$

closed form solutions

- Hessian is a $N \times N \times N \times N$ 4th order tensor.

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_j y_l]$$

[Cardoso & Laheld 1996]

(relative) Newton method

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_j y_l]$$

One iteration: $$W_{n+1} = (I - H^{-1}G)W_n$$

Problem:

Large linear system / not positive (regularization needed)

Newton method is possible but not practical

(relative) Newton method

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_j y_l]$$

One iteration: $$W_{n+1} = (I - H^{-1}G)W_n$$

Problem:

Large linear system / not positive (regularization needed)

Newton method is possible but not practical

but if model holds:

$$\delta_{ik}E[\psi_i'(y_i)y_j y_l] \;=\; \delta_{ik}\delta_{jl}E[\psi_i'(y_i)]E[y_j^2]$$

Hessian approximations

True hessian:

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_j y_l]$$

- If we **suppose that the signals are independent**
- Two Hessian approximations :

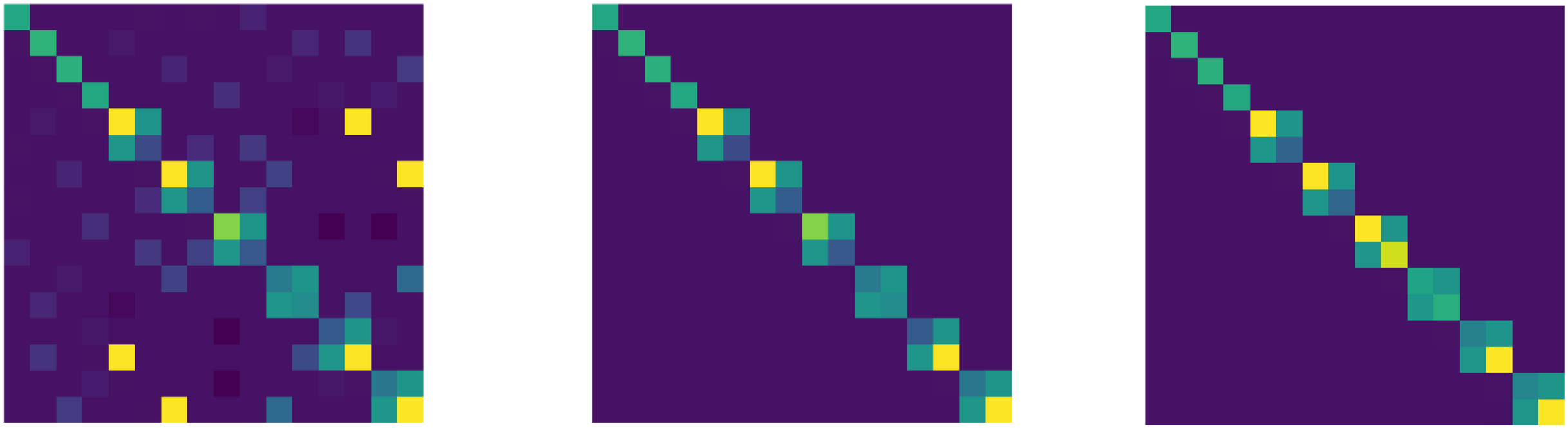
Complexity:

$$H^2_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}E[\psi_i'(y_i)y_j^2] \qquad \text{O(T N}^2\text{)}$$

$$H^1_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}E[\psi_i'(y_i)]E[y_j^2] \qquad \text{O(T N)}$$

Hessian approximations (cont.)

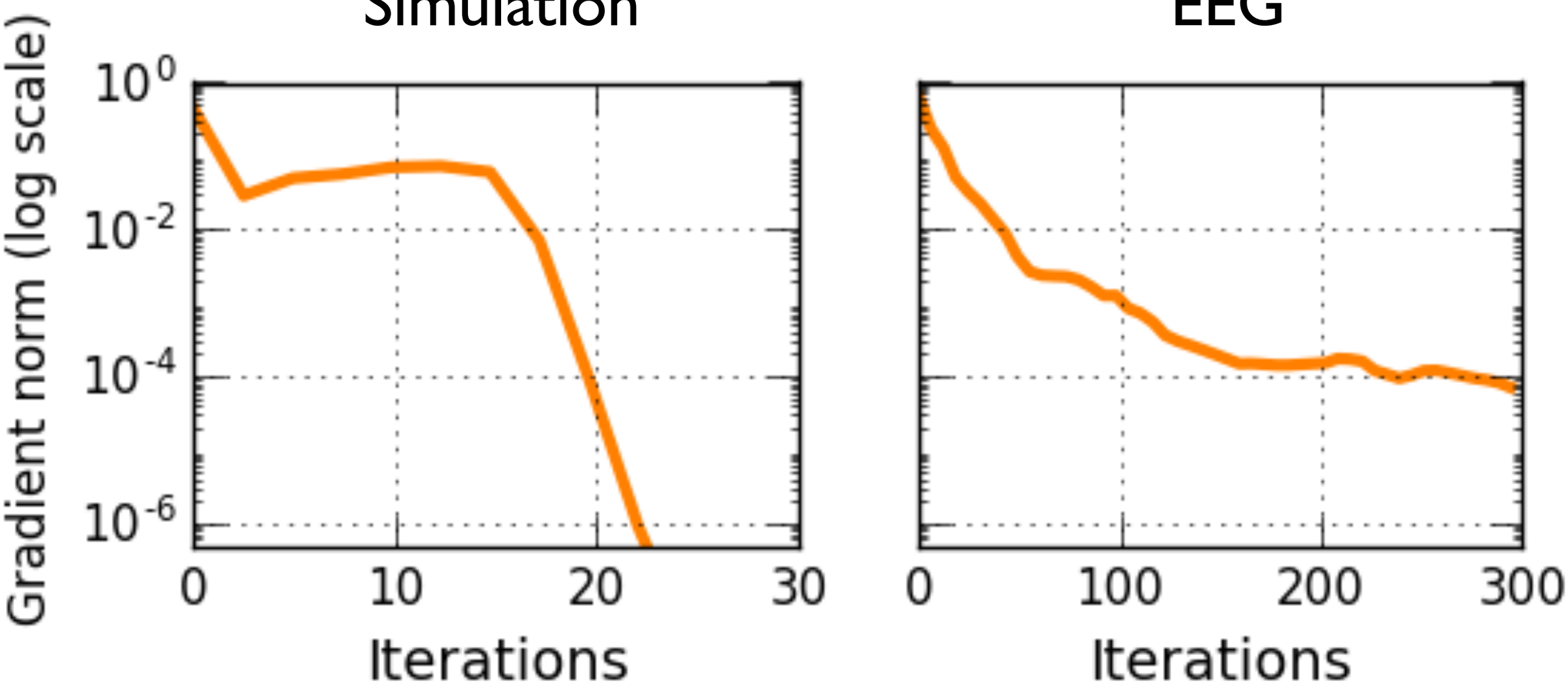
H H^2 H^1



- Hessians approximations are block-diagonal hence much easier to invert
- **Idea:**

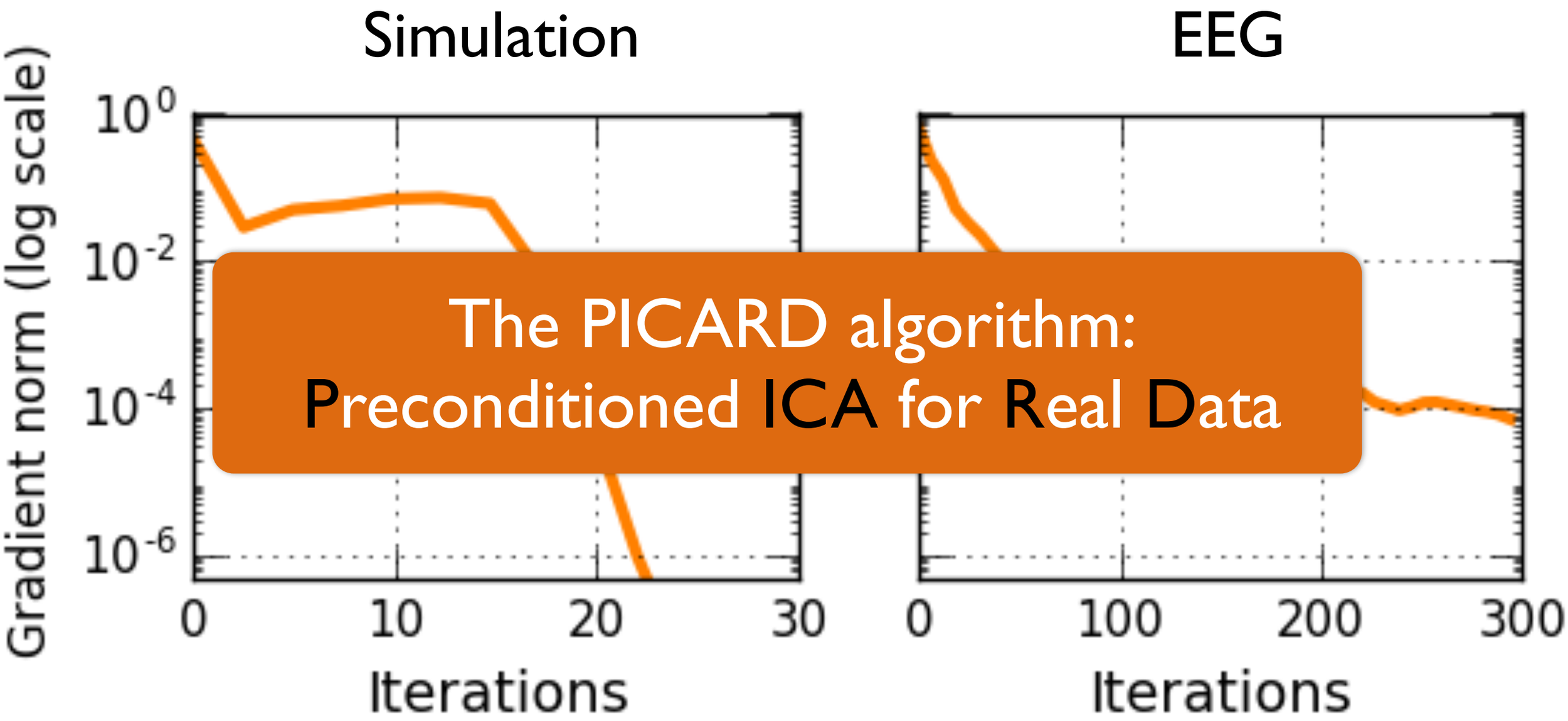
Do Newton with the approximated Hessian !

But…



It does not work with real data !

But…



The PICARD algorithm:
Preconditioned ICA for Real Data

It does not work with real data !

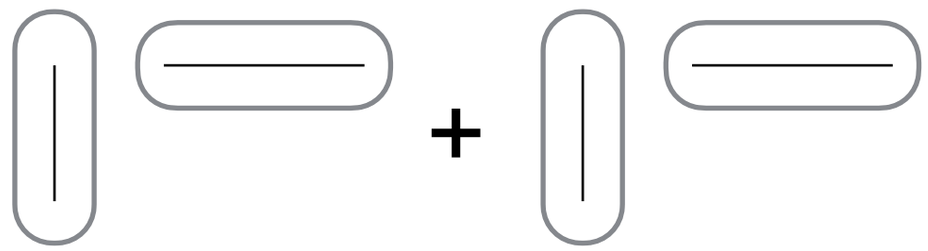
Can these guys help us retrieve missing information about curvature without computing the full Hessian ?

- C. G. Broyden, « The Convergence of a Class of Double-rank Minimization Algorithms », Journal of the Institute of Mathematics and Its Applications, 1970
- R. Fletcher, « A New Approach to Variable Metric Algorithms », Computer Journal, 1970
- D. Goldfarb, « A Family of Variable Metric Updates Derived by Variational Means », Mathematics of Computation, 1970
- D. F. Shanno, « Conditioning of Quasi-Newton Methods for Function Minimization », Mathematics of Computation, 1970

Picard relies on L-BFGS algorithm

- BFGS builds an approximation of the Hessian using only the past function and gradient evaluations

- Does low-rank corrections of an initial guess $H^{(0)}$ of the Hessian

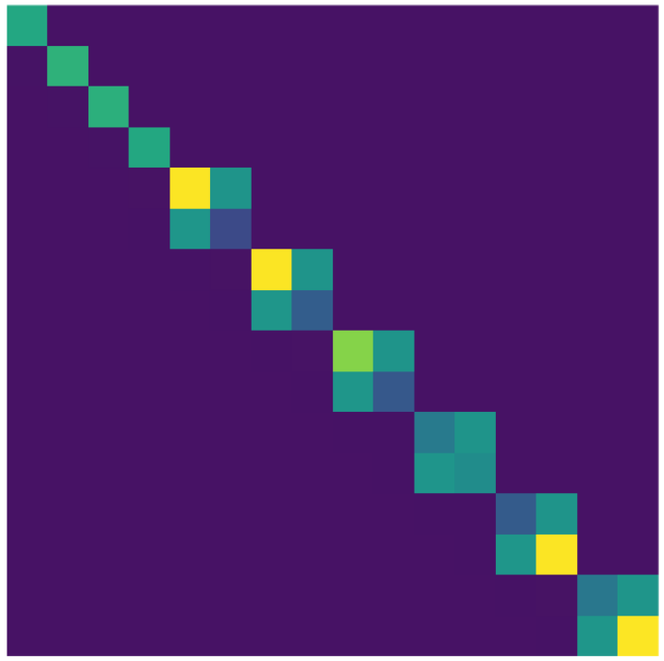
$H^{(q)}$ =  + $H^{(q-1)}$ (rank-2 updates)

- Initial guess $H^{(0)}$ should be easy to invert and is commonly a multiple of identity

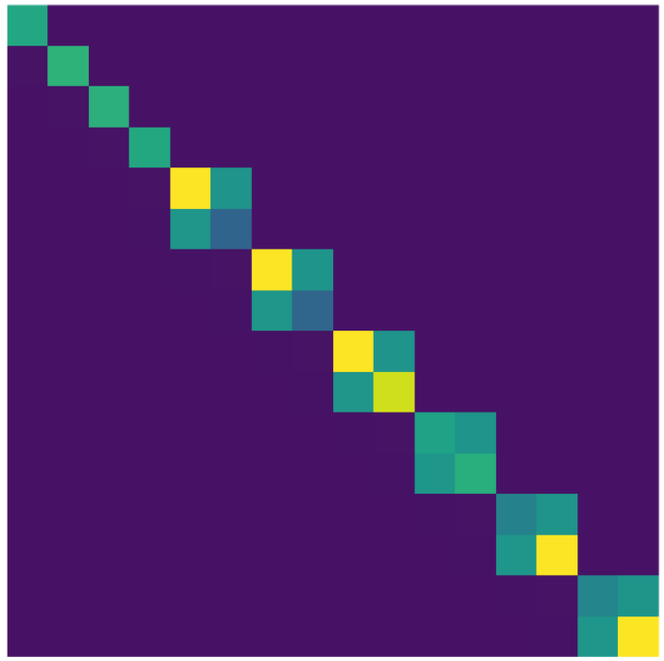
[Liu, D. C., & Nocedal, J. « On the limited memory BFGS method for large scale optimization. » *Mathematical programming, 1989]*

L-BFGS algorithm with Hessian approx.

H^2



H^1



Use these matrices as $H^{(0)}$ in L-BFGS

[P. Ablin, J.-F. Cardoso, A. Gramfort, Faster ICA by preconditioning with Hessian approximations, IEEE Trans. Signal Processing]

What are other state-of-the art solvers?

- Stochastic (relative) gradient a.k.a. Infomax

[Bell and Sejnowski 1995]

- Batch (relative) gradient descent
- Truncated Newton with full Hessian (using conjugate gradient solver and "Hessian free" products)

[Tillet et al. 2017]

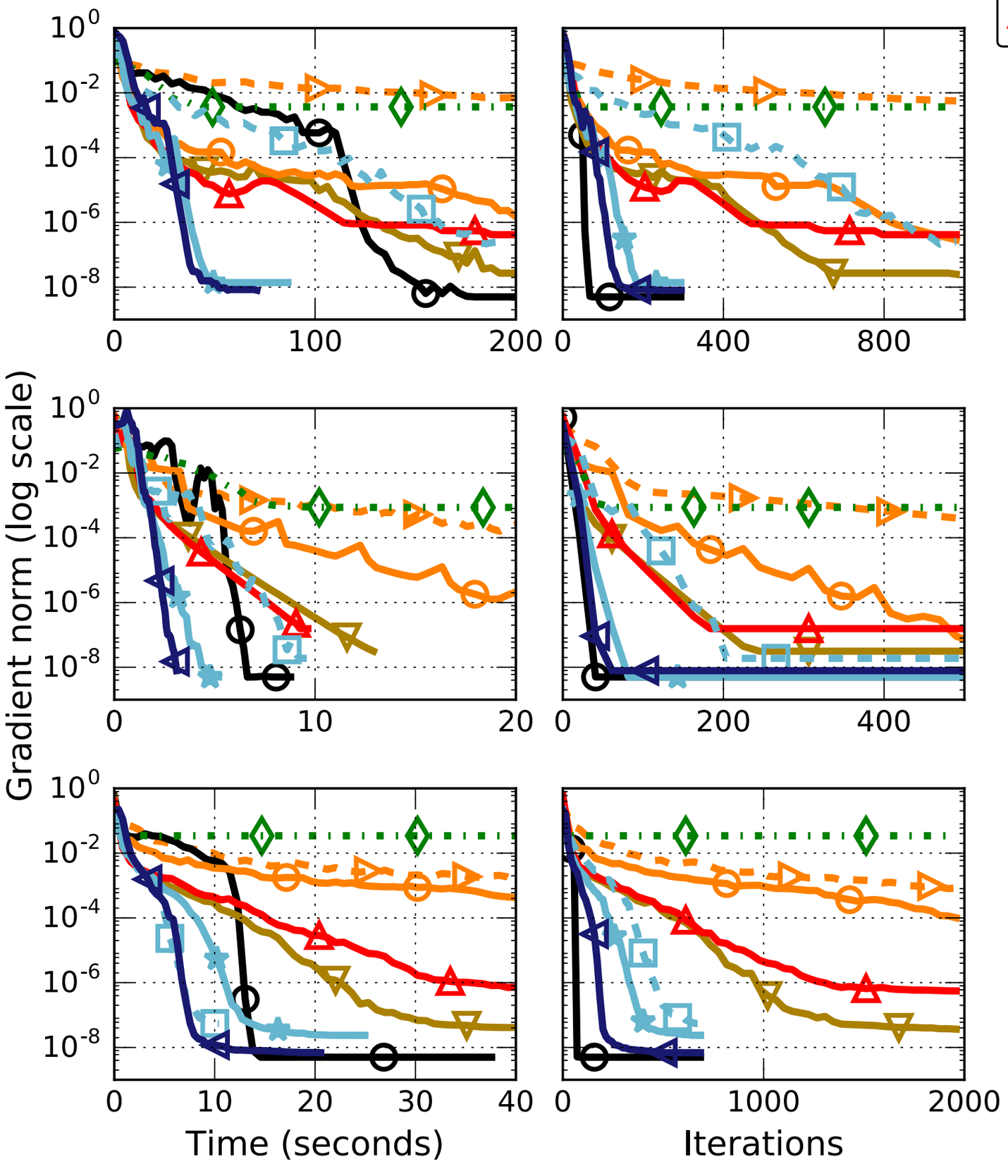
- Quasi-Newton with H1 or H2 Hessians

close to [Palmer et al. 2012]

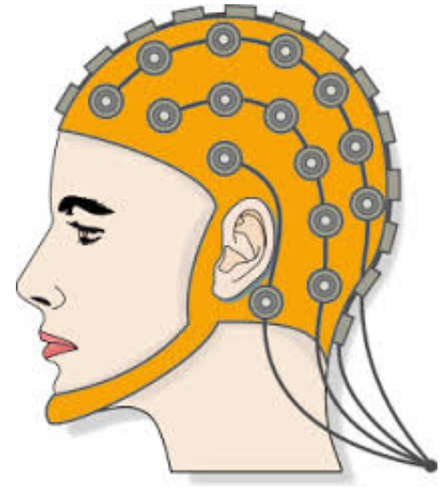
- Trust region ICA with H2 approximation

[Choi et al. 2007]

Real data



EEG



Functional MRI

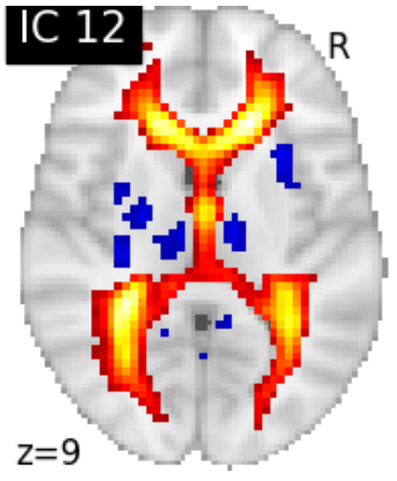
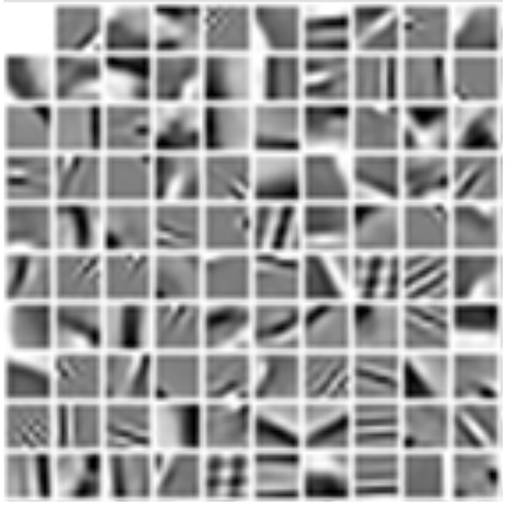


Image patches

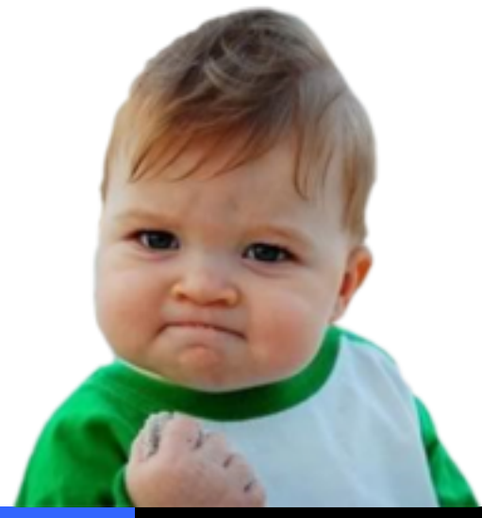


Can we reuse the Hessian approx. idea?

[P. Ablin, J.F. Cardoso and A. Gramfort.
Beyond Pham's algorithm for joint diagonalization.
Proc. ESANN 2019, https://hal.archives-ouvertes.fr/hal-01936887v1]

[P. Ablin, D. Fagot, H. Wendt, A. Gramfort and C. Févotte.
A Quasi-Newton algorithm on the orthogonal manifold for NMF with transform learning.
Proc. ICASSP 2019, https://hal.archives-ouvertes.fr/hal-01912918/document]

[P. Ablin, J.F. Cardoso and A. Gramfort.
Faster ICA under orthogonal constraint.
Proc. ICASSP'18, https://hal.inria.fr/hal-01651842/document]



Problem 1: Picard is a batch method

Problem 2: SGD (here Infomax)

- has no descent guarantee
- requires careful tuning of learning rate

[EM algorithms for ICA, P. Ablin, A. Gramfort, J.F. Cardoso and F. Bach, AISTATS 2019]

Problem 1: Picard is a batch method

Problem 2: SGD (here Infomax)

- has no descent guarantee
- requires careful tuning of learning rate

Question: Can we find an online algorithm not more costly than SGD with descent guarantee and no learning rate?

[EM algorithms for ICA, P. Ablin, A. Gramfort, J.F. Cardoso and F. Bach, AISTATS 2019]

ICA, EM and quadratic majorant

Assumption: $p_i \overset{\text{def}}{=} d$ Same density per source

We denote: $G(y) = -\log(d(y))$

Objective function:

$$\mathcal{L}(W) = -\log|\det(W)| + \frac{1}{T}\sum_{i=1}^{N}\sum_{j=1}^{T} G([WX]_{ij})$$

ICA, EM and quadratic majorant

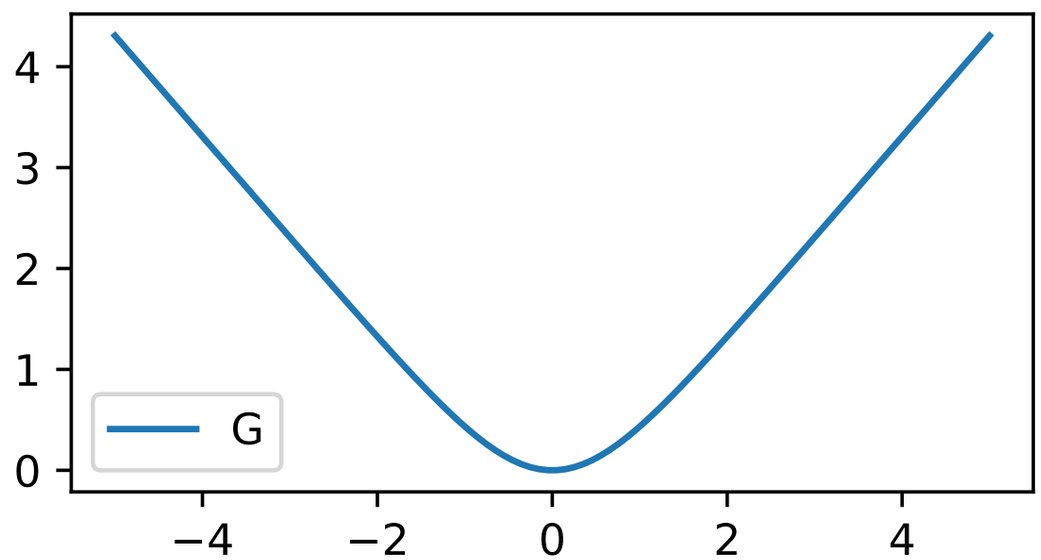
Assumption: $p_i \overset{\text{def}}{=} d$ Same density per source

We denote: $G(y) = -\log(d(y))$

Objective function:

$$\mathcal{L}(W) = -\log|\det(W)| + \frac{1}{T}\sum_{i=1}^{N}\sum_{j=1}^{T} G([WX]_{ij})$$

Key assumption: $G(\sqrt{\cdot})$ is concave (super-Gaussian source)

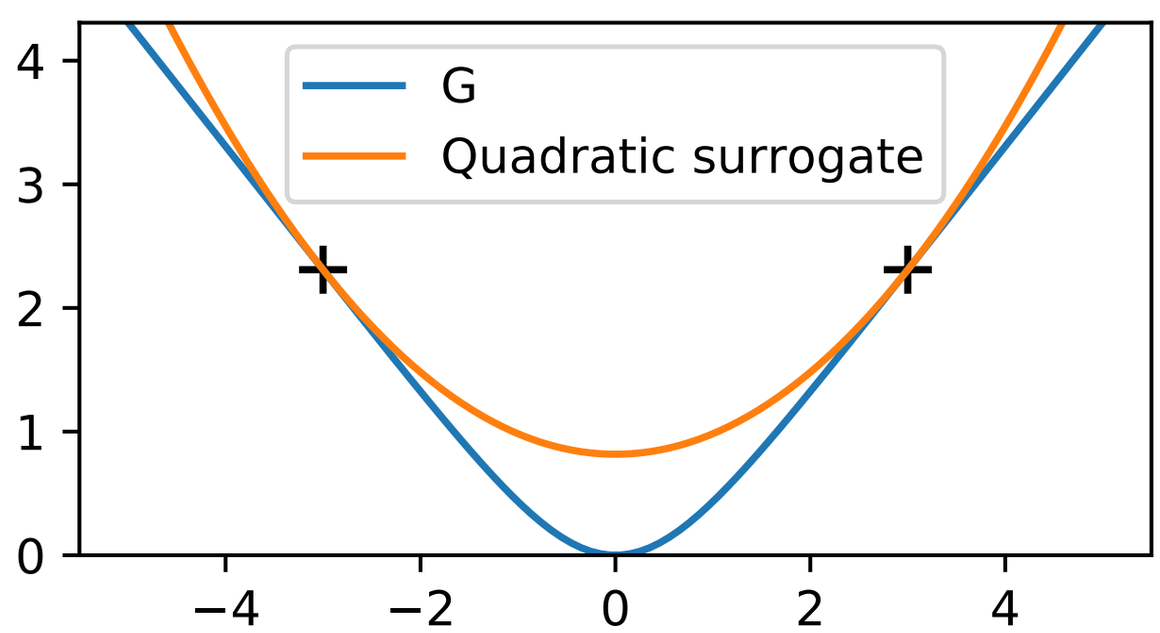


ICA, EM and quadratic majorant

If $G(\sqrt{\cdot})$ is concave (super-Gaussian source)

There exists a function f such that:

$$G(y) = \min_{u \geq 0} \frac{uy^2}{2} + f(u) \quad \text{(cf. Fenchel duality)}$$



Minimum is reached at $u^*(y) = \dfrac{G'(y)}{y}$

ICA, EM and quadratic majorant

The new function reads:

$$\mathcal{L}(W, U) =$$

$$-\log|\det(W)| + \frac{1}{2T}\sum_{i=1}^{N}\sum_{j=1}^{T} U_{ij}\,[WX]_{ij}^2 + \frac{1}{T}\sum_{i=1}^{N}\sum_{j=1}^{T} f(U_{ij})$$

[EM algorithms for ICA, P. Ablin, A. Gramfort, J.F. Cardoso and F. Bach, AISTATS 2019]

ICA, EM and quadratic majorant

The new function reads:

$$\mathcal{L}(W, U) =$$

$$-\log|\det(W)| + \frac{1}{2T}\sum_{i=1}^{N}\sum_{j=1}^{T} U_{ij}\,[WX]_{ij}^2 + \frac{1}{T}\sum_{i=1}^{N}\sum_{j=1}^{T} f(U_{ij})$$

One can minimize by alternate minimization

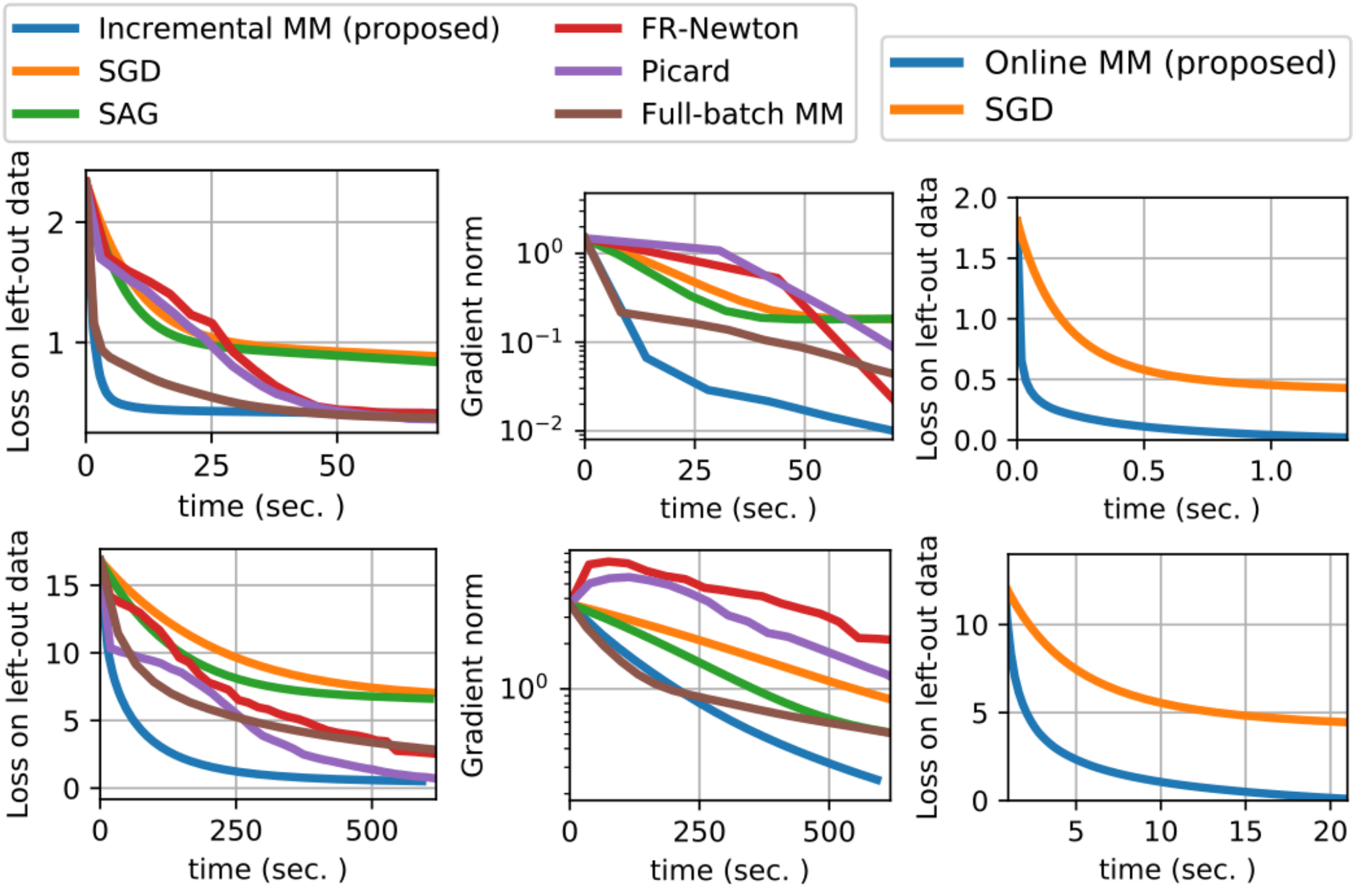
… using ideas from variance reduced stochastic methods (eg. SAG)

… with memory and some accumulation of the right quantities

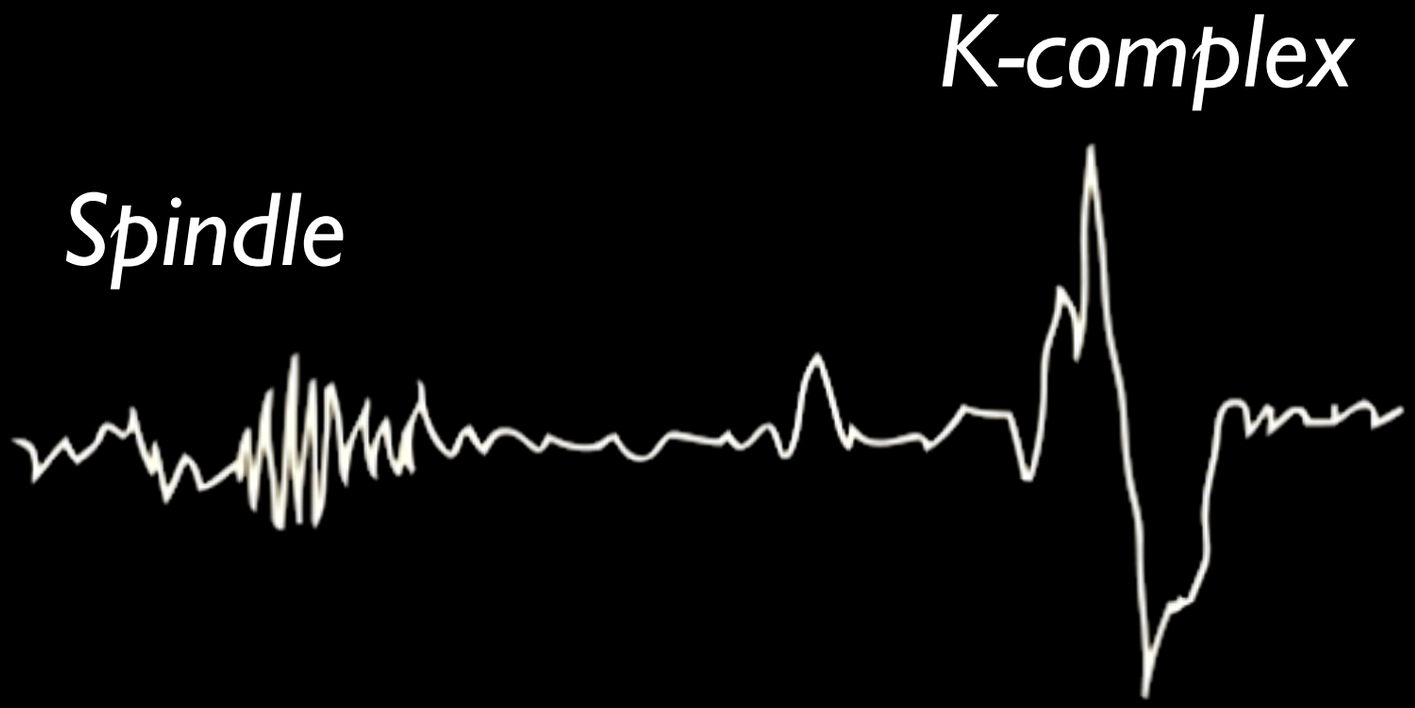
… and carefully coordinate descent steps

[EM algorithms for ICA, P. Ablin, A. Gramfort, J.F. Cardoso and F. Bach, AISTATS 2019]

Experiments on real data



Learning patterns / waveforms from (multivariate) signals



[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

[Distributed Convolutional Dictionary Learning (DiCoDiLe): Pattern Discovery in Large Images and Signals (2019), T. Moreau and A. Gramfort, ArXiv.]

Code: https://alphacsc.github.io

Learning patterns / waveforms from (multivariate) signals

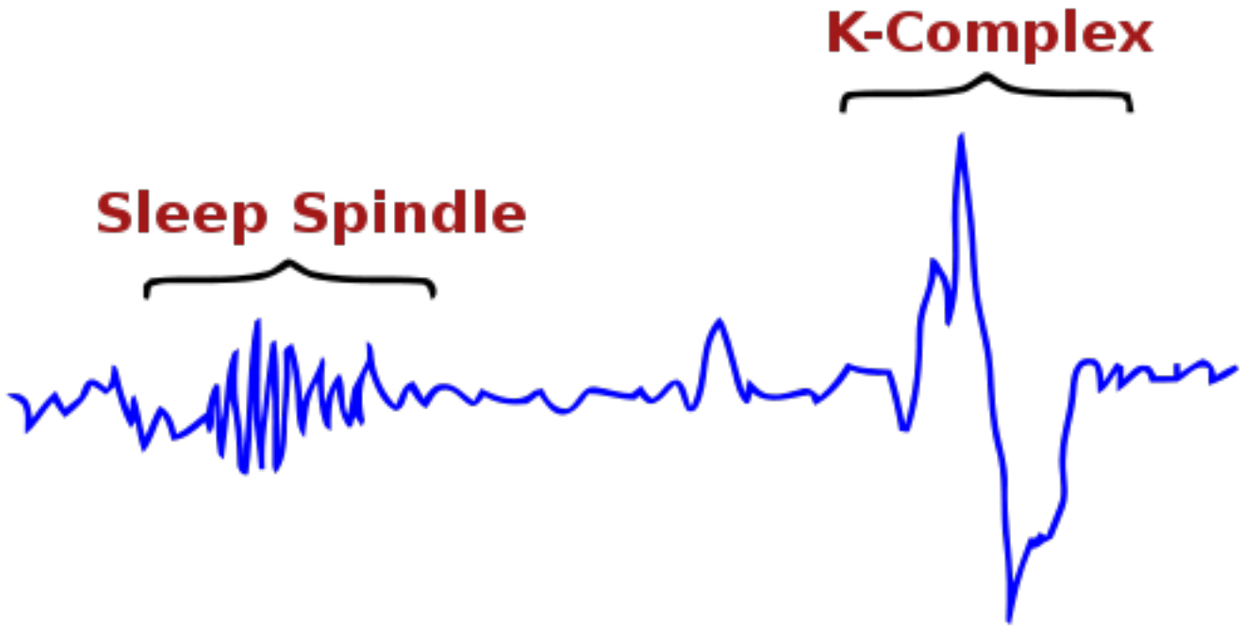
Objective:
Physics informed model
Fast solvers

[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

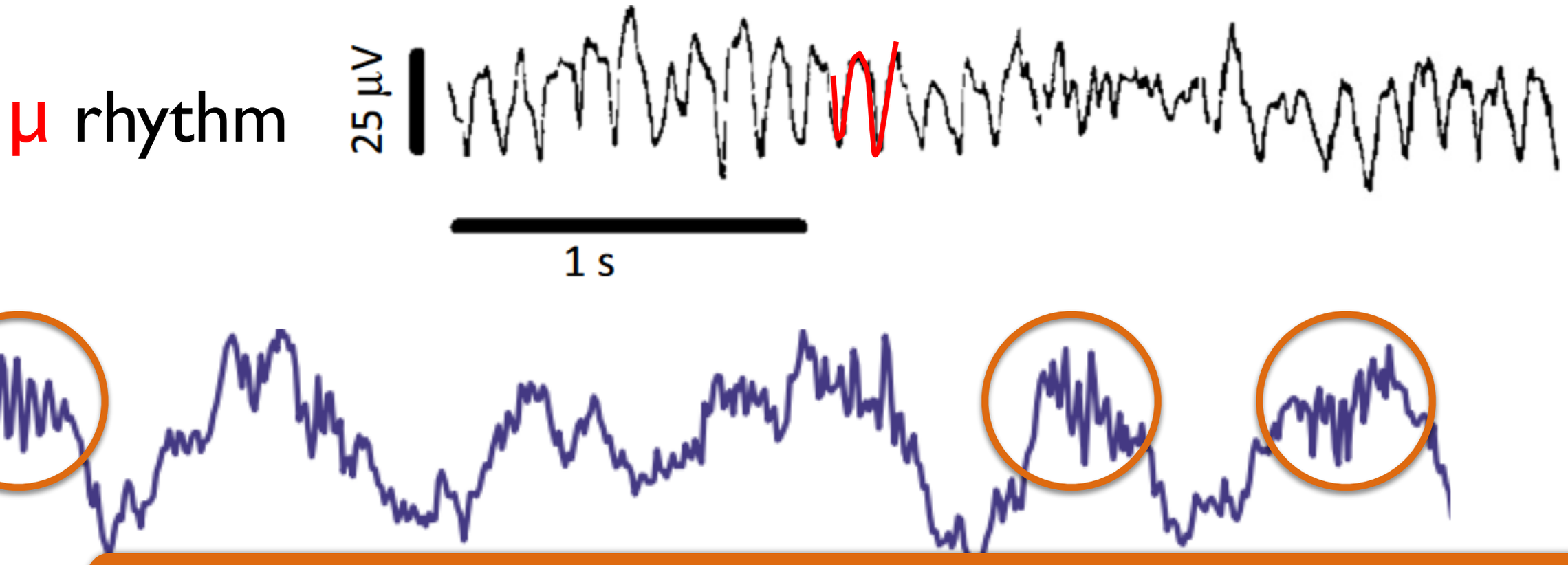
[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

[Distributed Convolutional Dictionary Learning (DiCoDiLe): Pattern Discovery in Large Images and Signals (2019), T. Moreau and A. Gramfort, ArXiv.]

Code: https://alphacsc.github.io



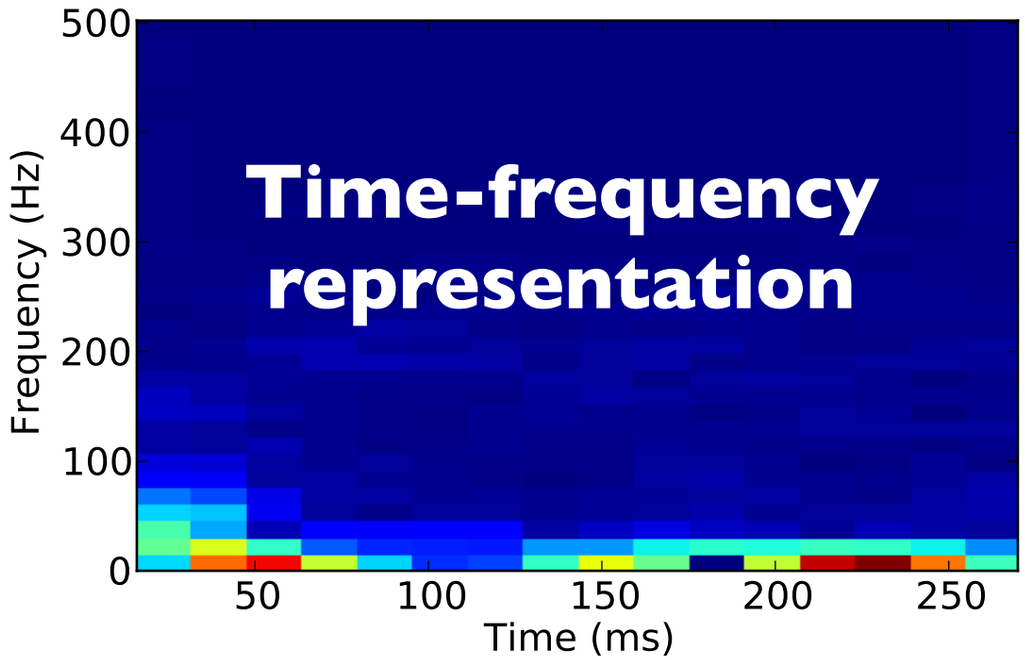
Neural signals exhibit diverse and complex morphologies

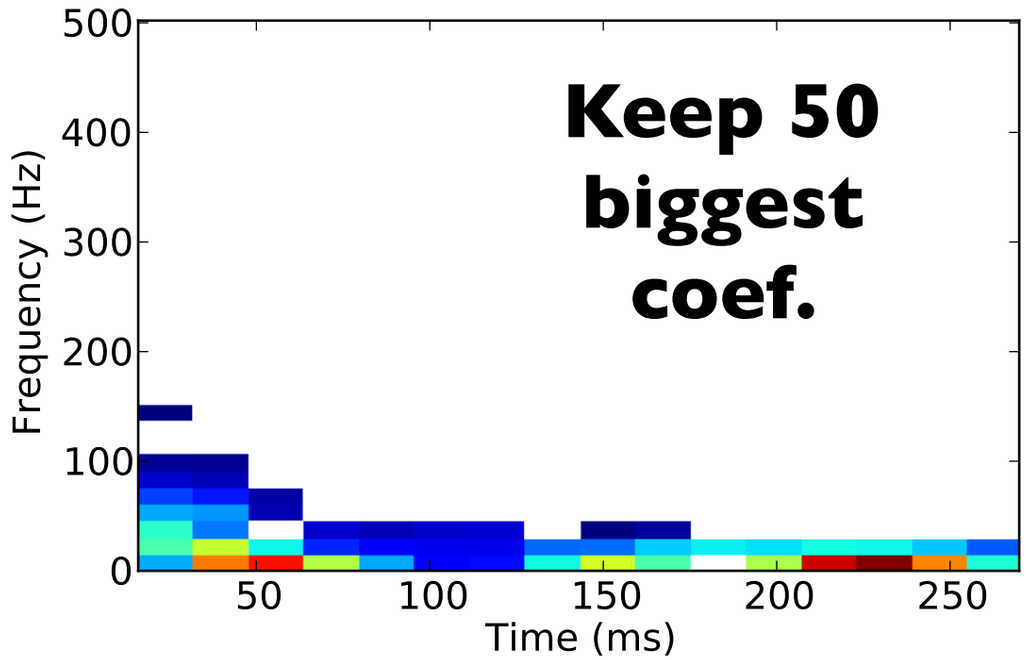


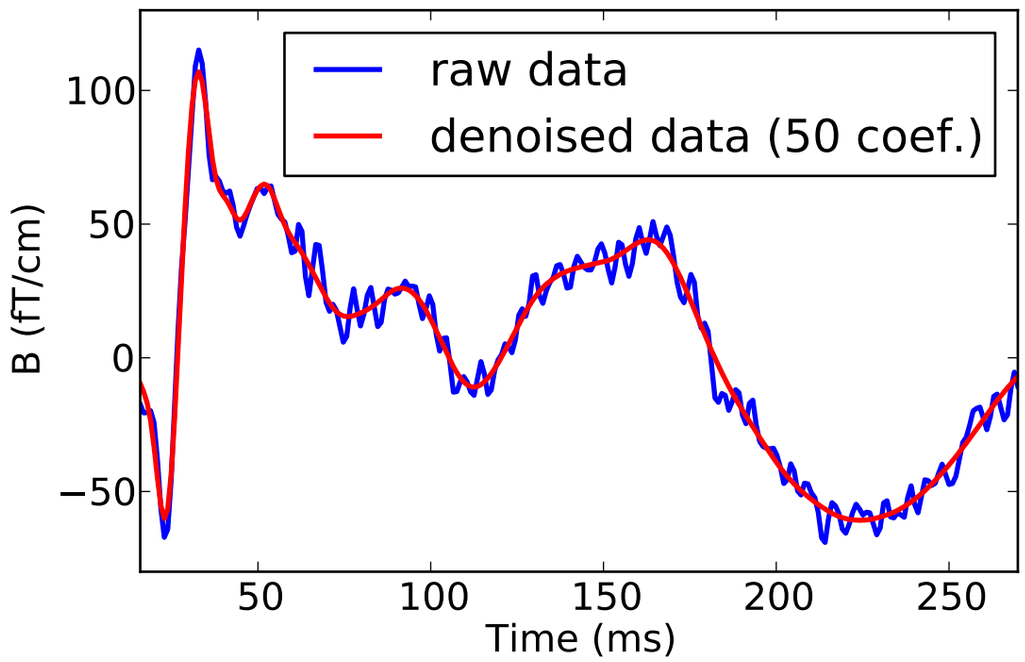
CFC: High frequency bursts coupled with slow waves

[T. Dupré la Tour, L. Tallot, L. Grabot, V. Doyère, V. van Wassenhove, Y. Grenier, A. Gramfort, (2017) PLOS Computational biology]

Why signal representations?







Having adequate signal representation help: denoising, prediction, inverse problems etc.

Signal representations

- Sparse representations: wavelet basis

[Morlet 70', Meyer 80', Mallat 90' etc.]

- Sparse coding / dictionary learning

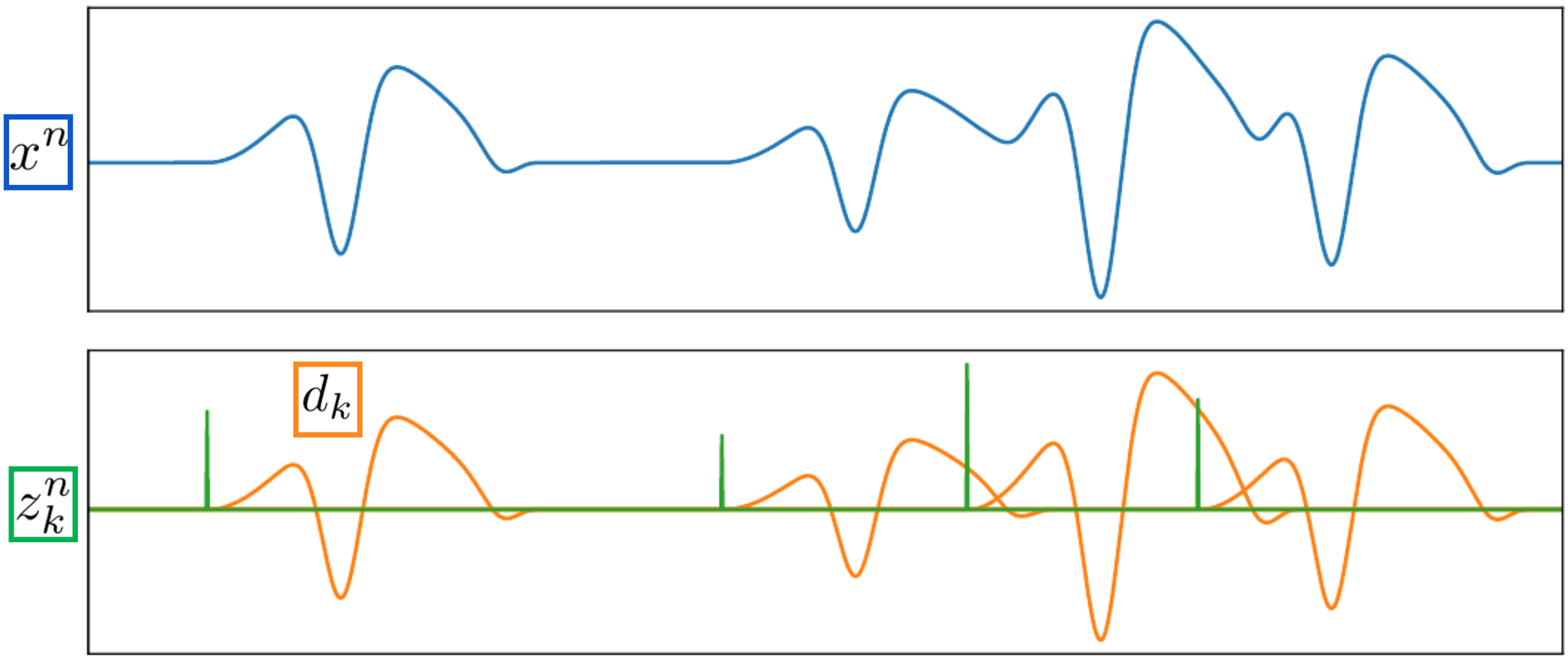
[Olshausen and Field, 1996, Elad and Aharon, 2006]

- Shift-invariant representations

[Lewicki and Sejnowski, 1999, Grosse et al, 2007]

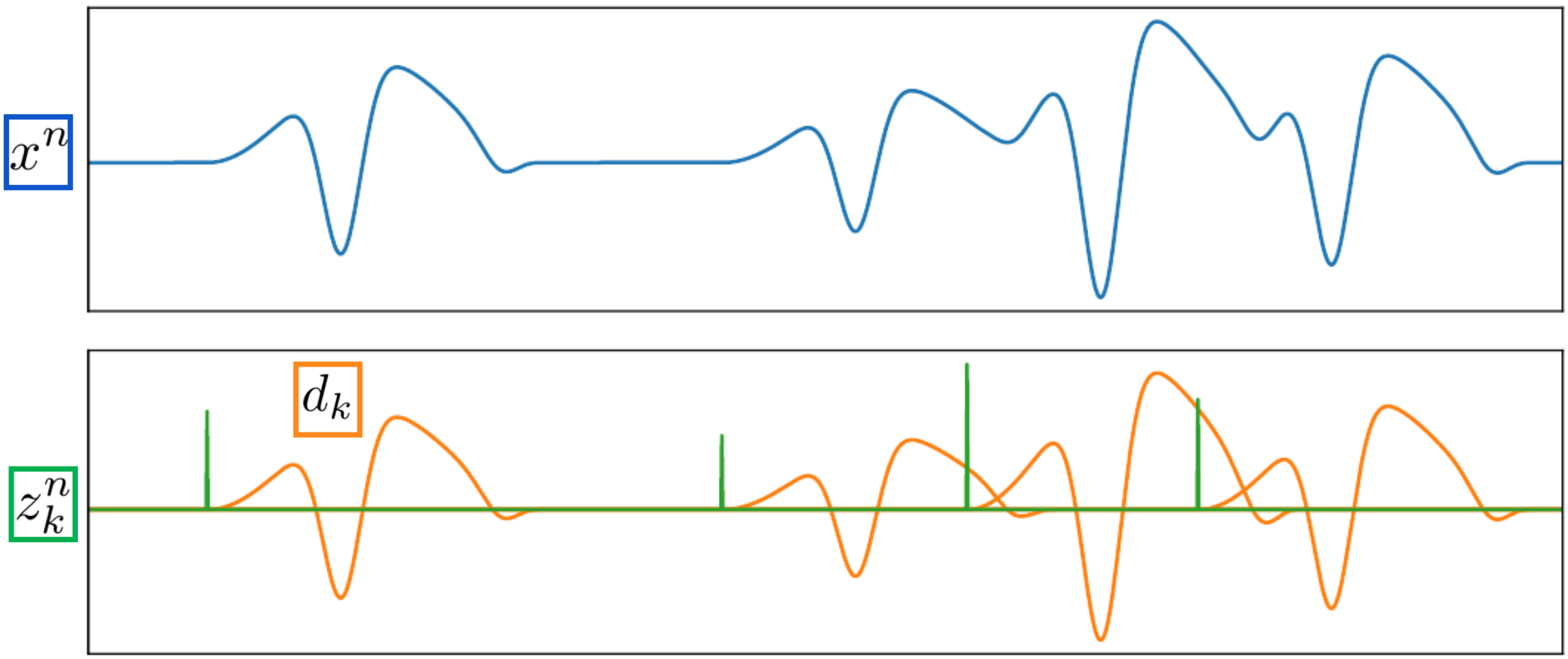
- In neurophysiology:
  - Matching of time-invariant filters (MOTIF) [Jost et al, 2006]
  - Multivariate orthogonal matching pursuit [Barthélemy et al, 2012]
  - Matching pursuit and heuristics [Brokmeier and Principe, 2016]
  - Sliding window machine [Gips et al, 2017]
  - Adaptive waveform learning [Hitziger et al, 2017]

Convolutional sparse coding



[Grosse et al, 2007]

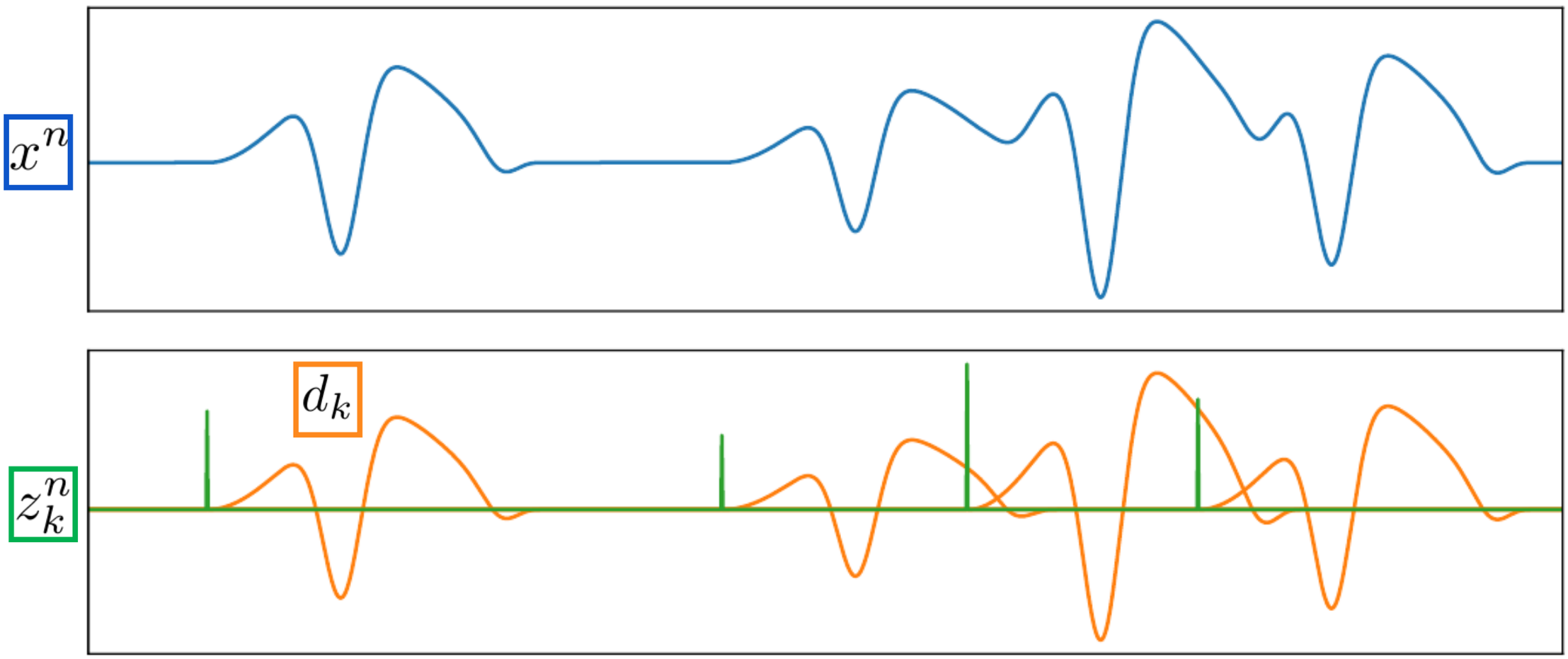
Convolutional sparse coding



$$x^n[t] = \sum_{k=1}^{K} (z_k^n * d_k)[t] + \varepsilon[t]$$

[Grosse et al, 2007]

Convolutional sparse coding



$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$

$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

[Grosse et al, 2007]

Optimization strategy

$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

Block-coordinate descent (update Z, D, Z, D, Z…):

Optimization strategy

$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

Block-coordinate descent (update Z, D, Z, D, Z…):

- Z-step
  - GCD [Kavukcuoglu et al, 2010]
  - FISTA [Chalasani et al, 2013]
  - ADMM [Bristow et al, 2013]
  - ADMM + FFT [Wohlberg, 2016]
  - L-BFGS [Jas et al, 2017]
  - LGCD [Dupré la Tour et al, 2018]

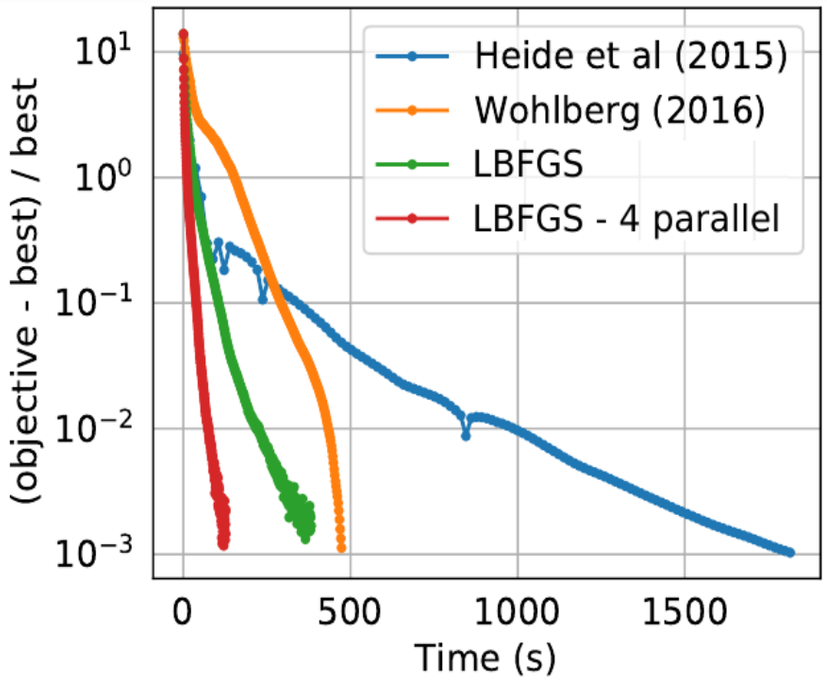
Optimization strategy

$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

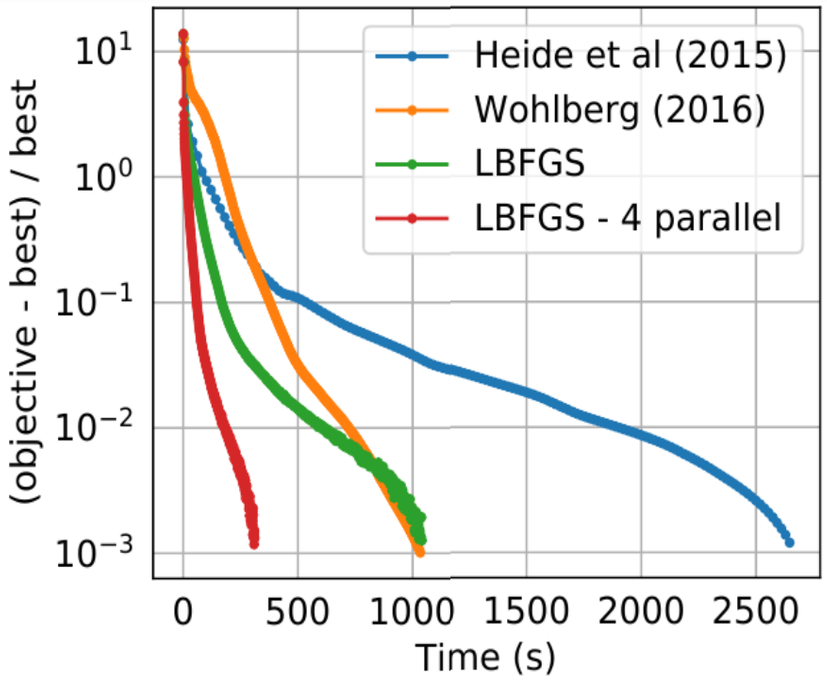
Block-coordinate descent (update Z, D, Z, D, Z…):

- Z-step
  - GCD [Kavukcuoglu et al, 2010]
  - FISTA [Chalasani et al, 2013]
  - ADMM [Bristow et al, 2013]
  - ADMM + FFT [Wohlberg, 2016]
  - L-BFGS [Jas et al, 2017]
  - LGCD [Dupré la Tour et al, 2018]
- D-step
  - FFT [Grosse et al, 2007]
  - ADMM + FFT [Heide et al, 2015]
  - ADMM + FFT [Wohlberg, 2016]
  - L-BFGS (dual) [Jas et al, 2017]
  - PGD [Dupré la Tour et al, 2018]

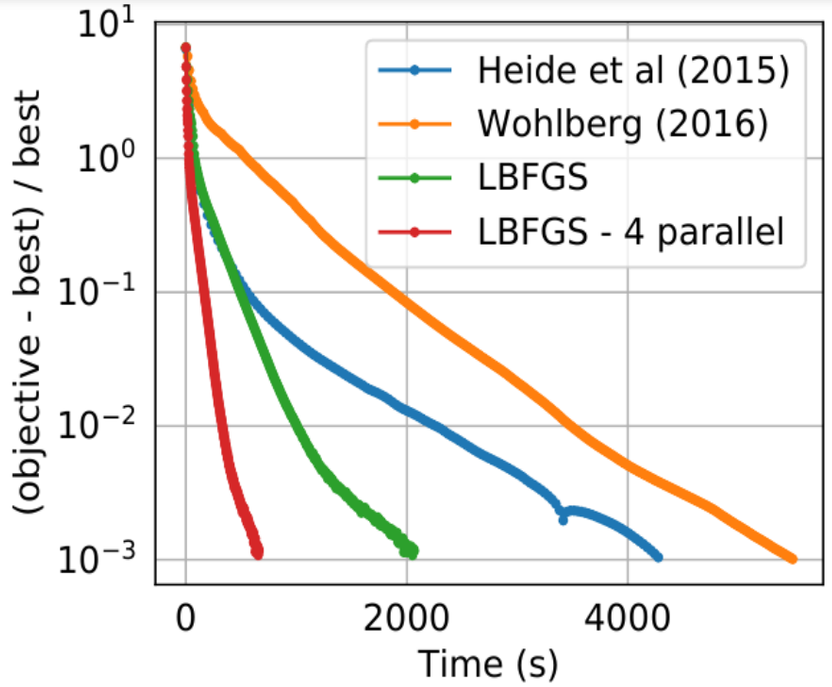
Speed benchmarks



(a) $K = 2$, $L = 32$.



(b) $K = 2$, $L = 128$.

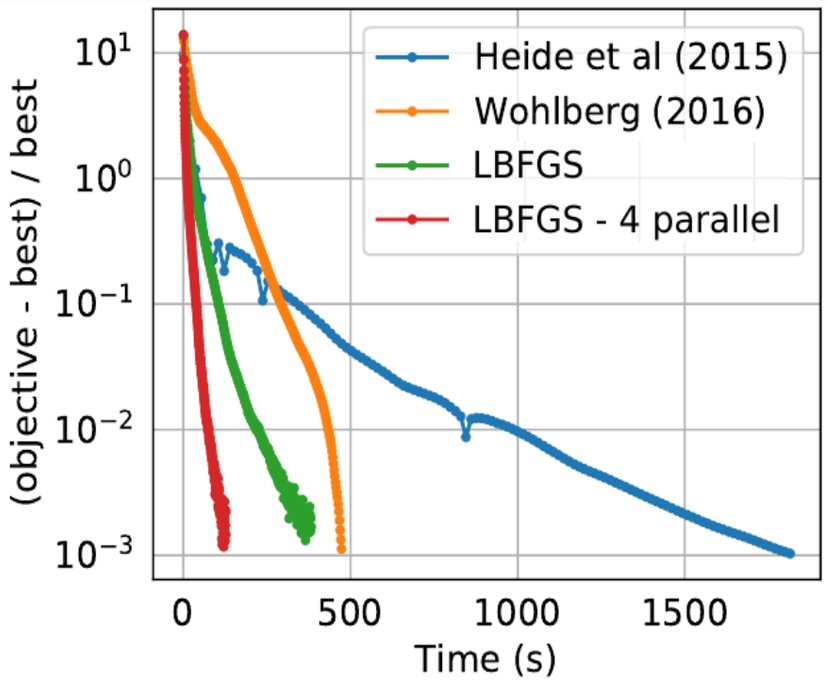


(c) $K = 10$, $L = 32$.

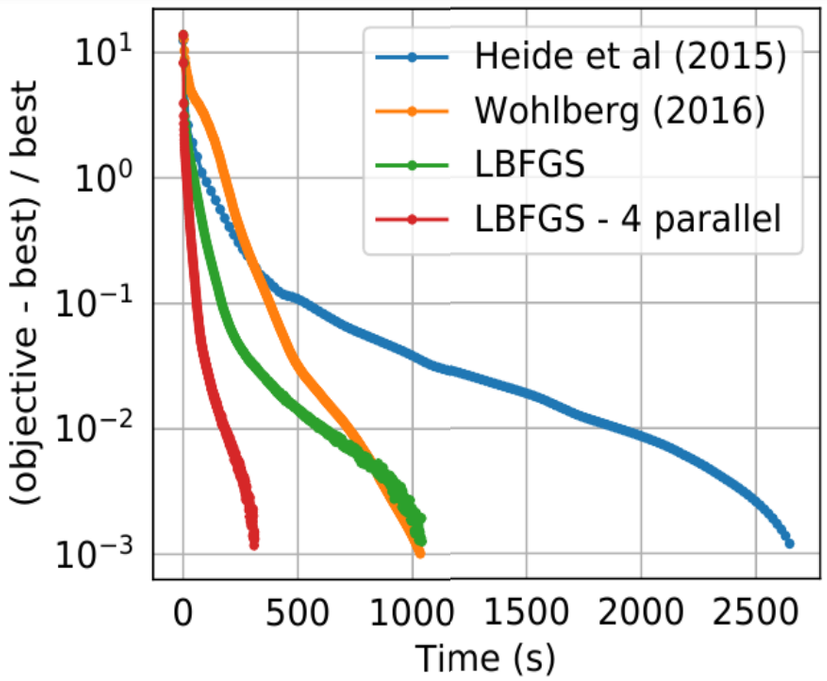
[Jas et al. 2017]

[Dupré la Tour et al. 2018]

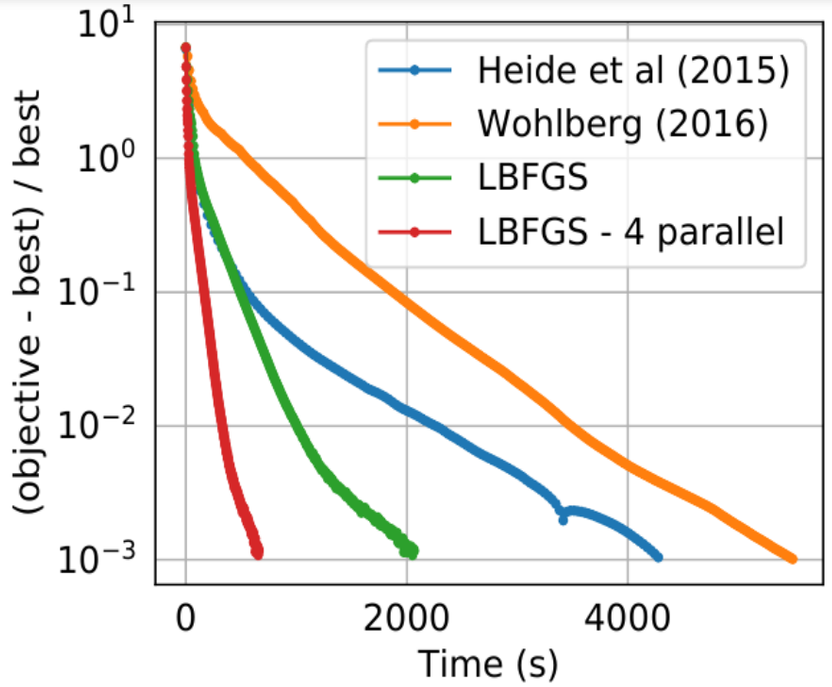
Speed benchmarks



(a) $K = 2$, $L = 32$.

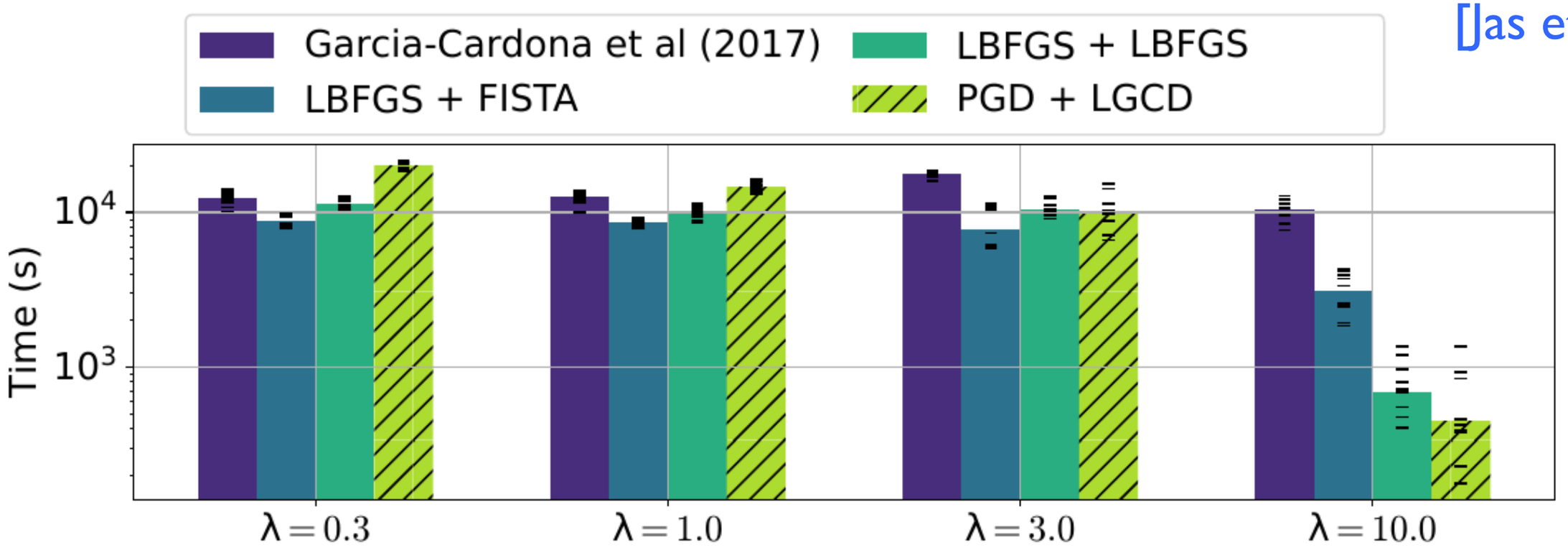


(b) $K = 2$, $L = 128$.



(c) $K = 10$, $L = 32$.

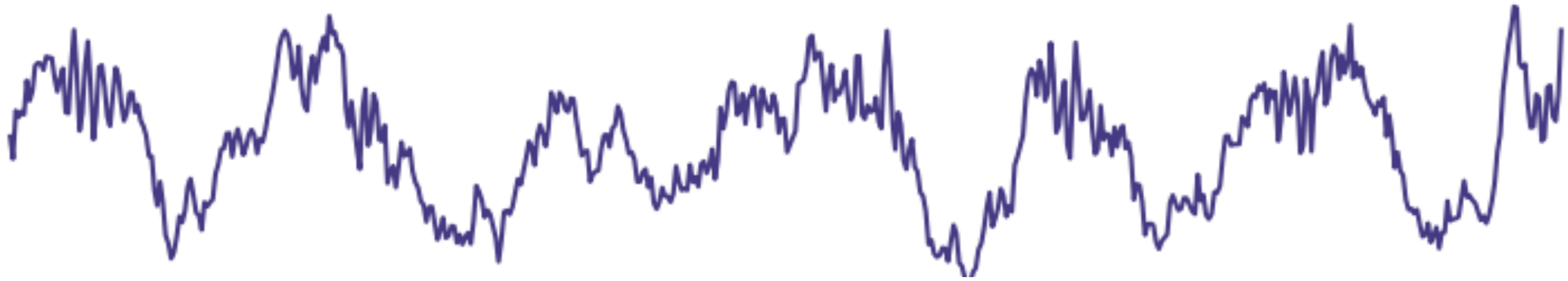
[Jas et al. 2017]

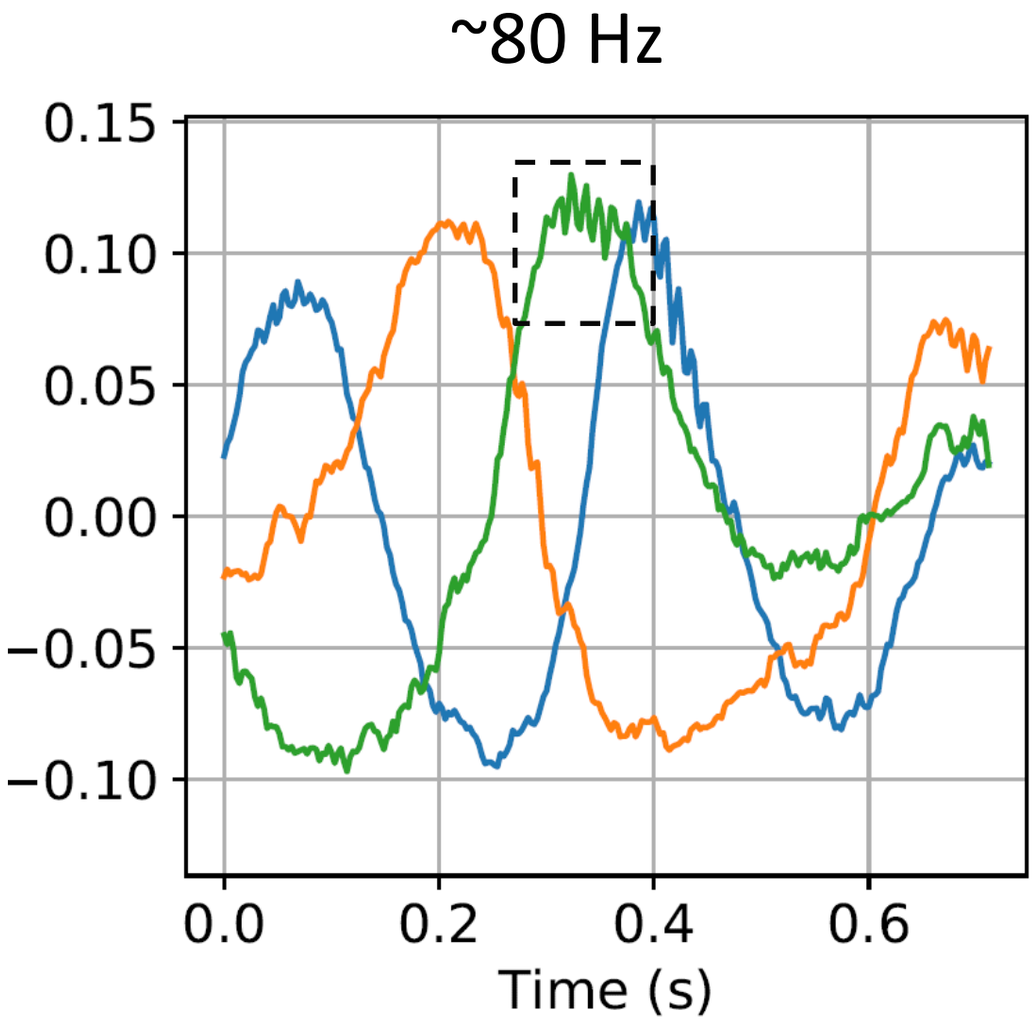


[Dupré la Tour et al. 2018]

Learned atoms

Data:

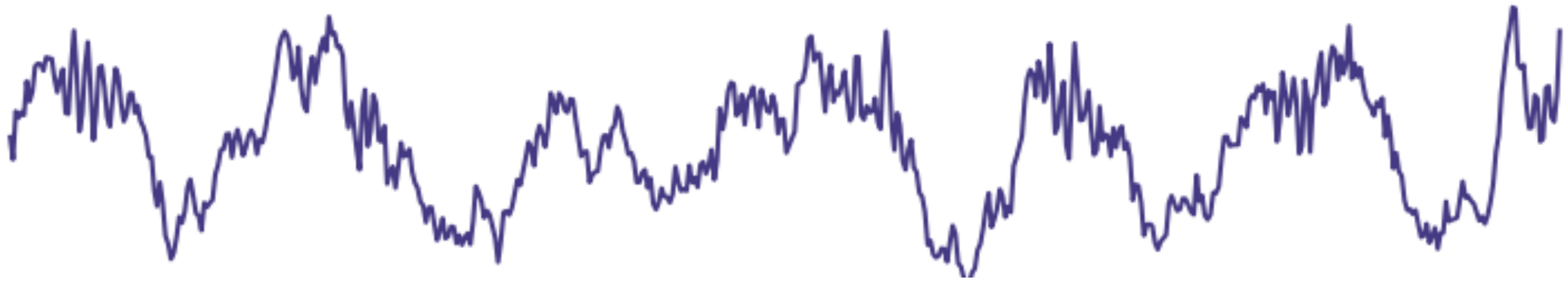




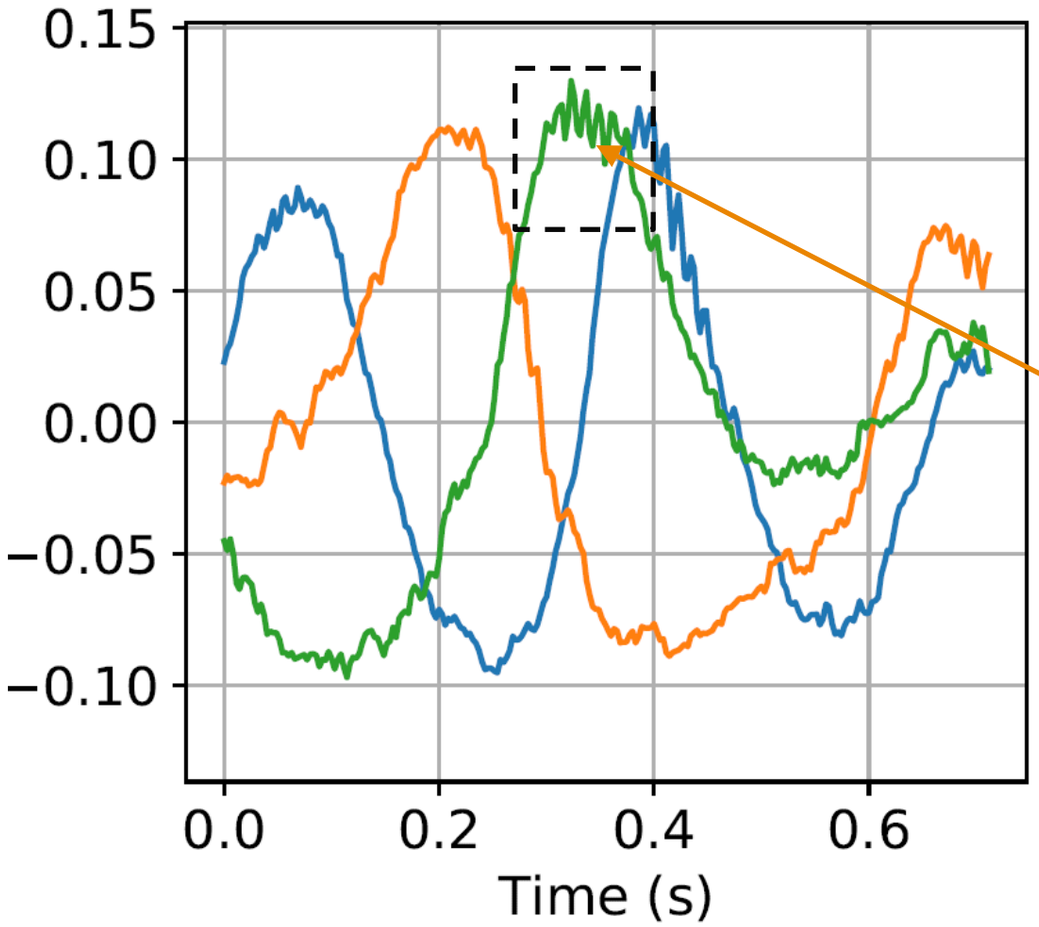
[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

Learned atoms

Data:



~80 Hz

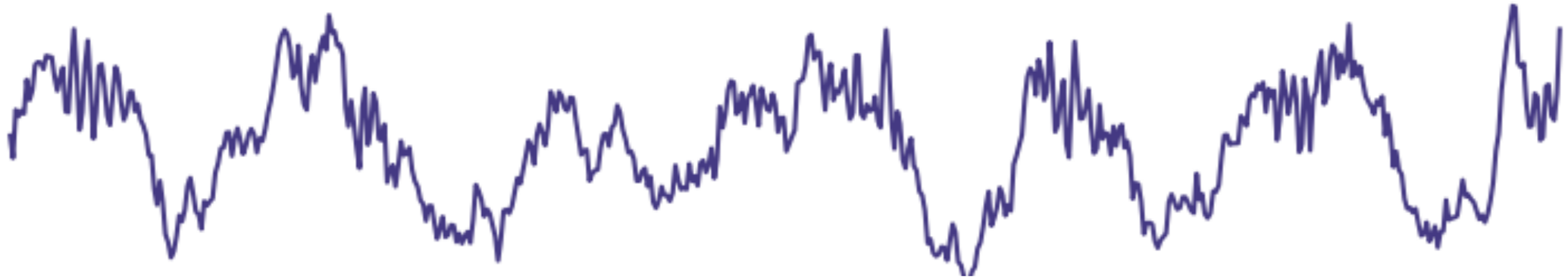


CSC reveals "nested oscillations"

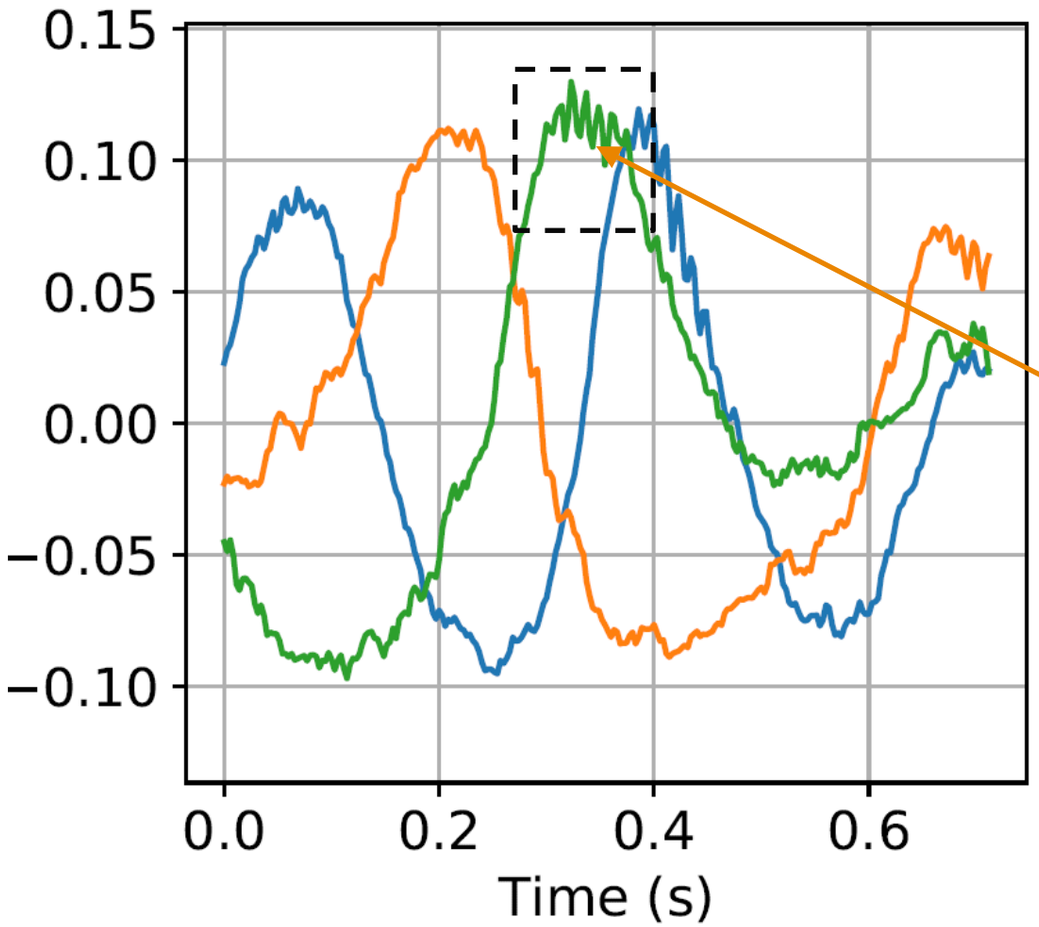
[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

Learned atoms

Data:



~80 Hz



How about if I have many channels?

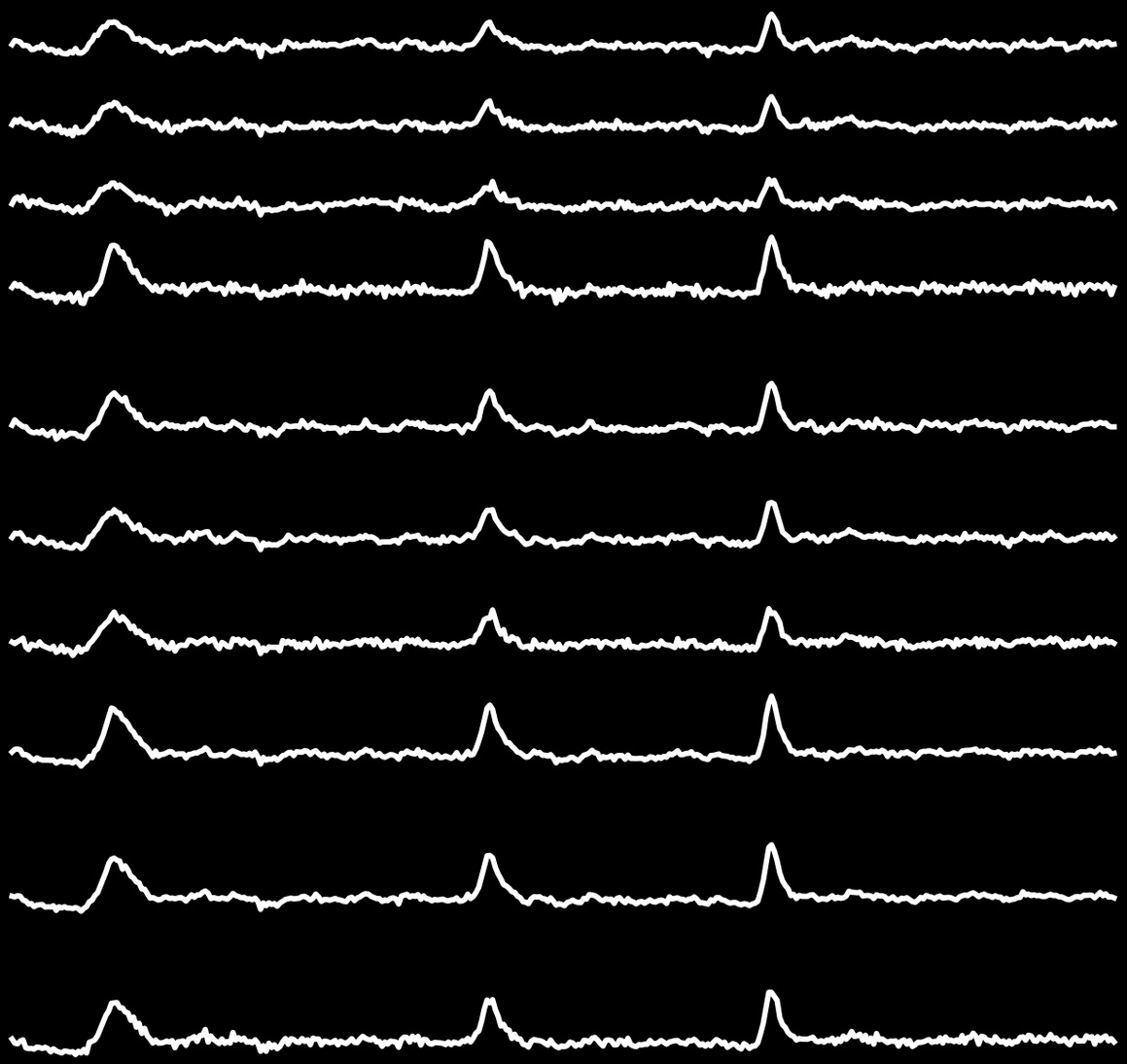
CSC reveals "nested oscillations"

[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

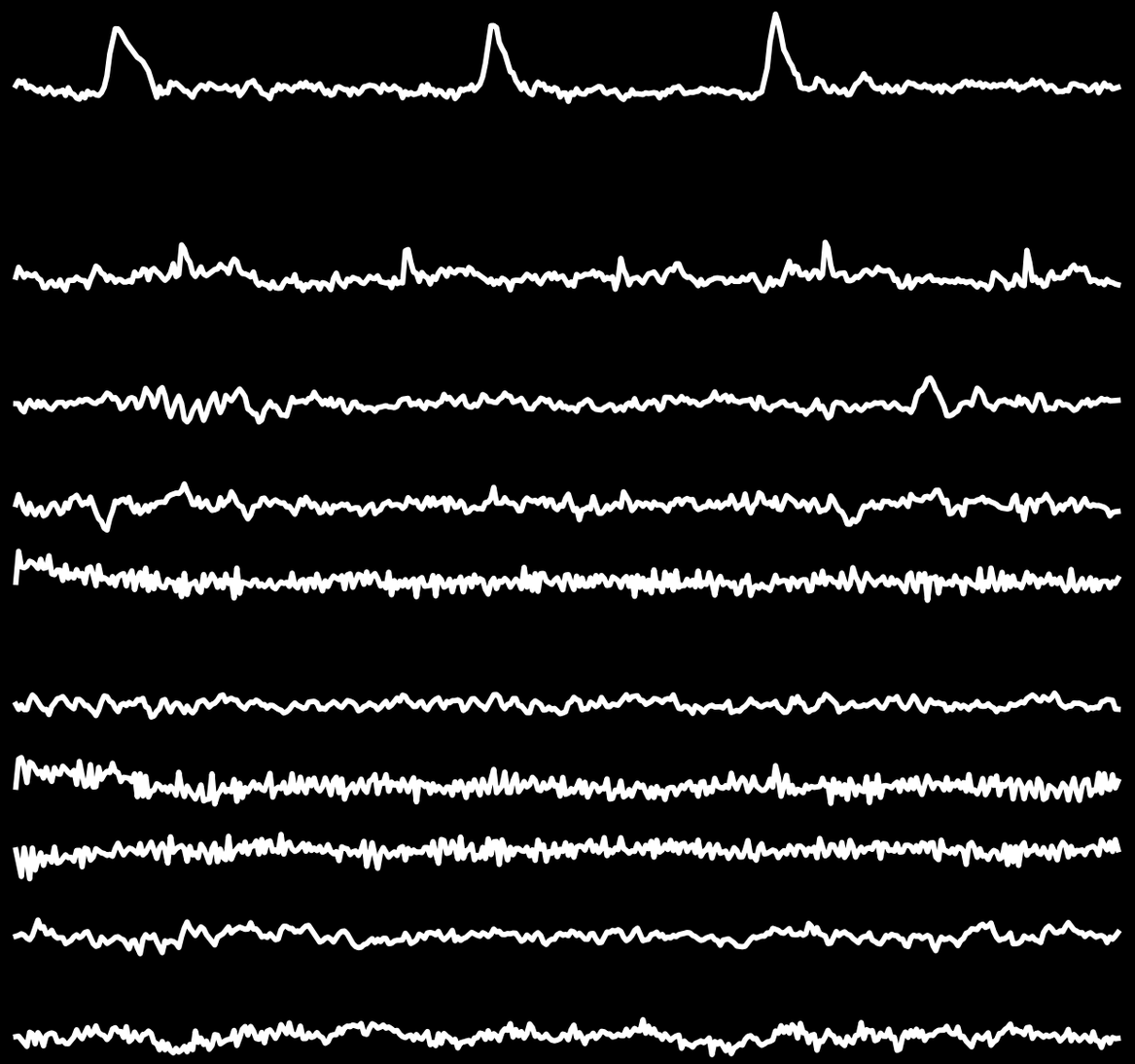
From ICA to CSC

Independent Component Analysis (ICA)

Observations (raw EEG)



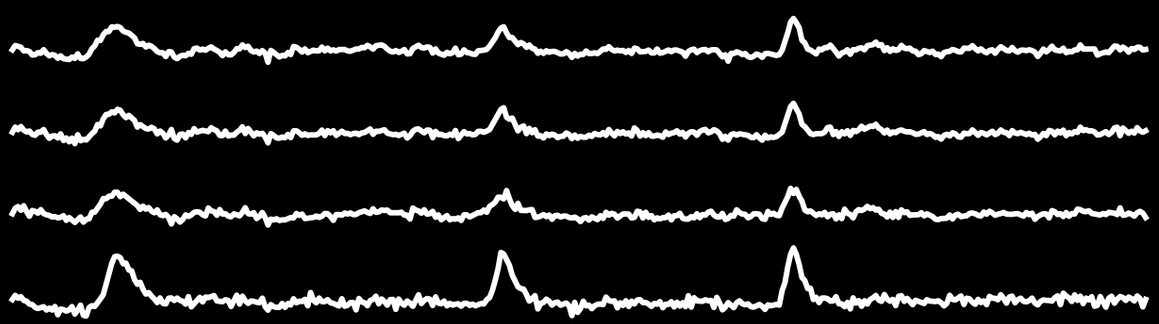
ICA recovered sources

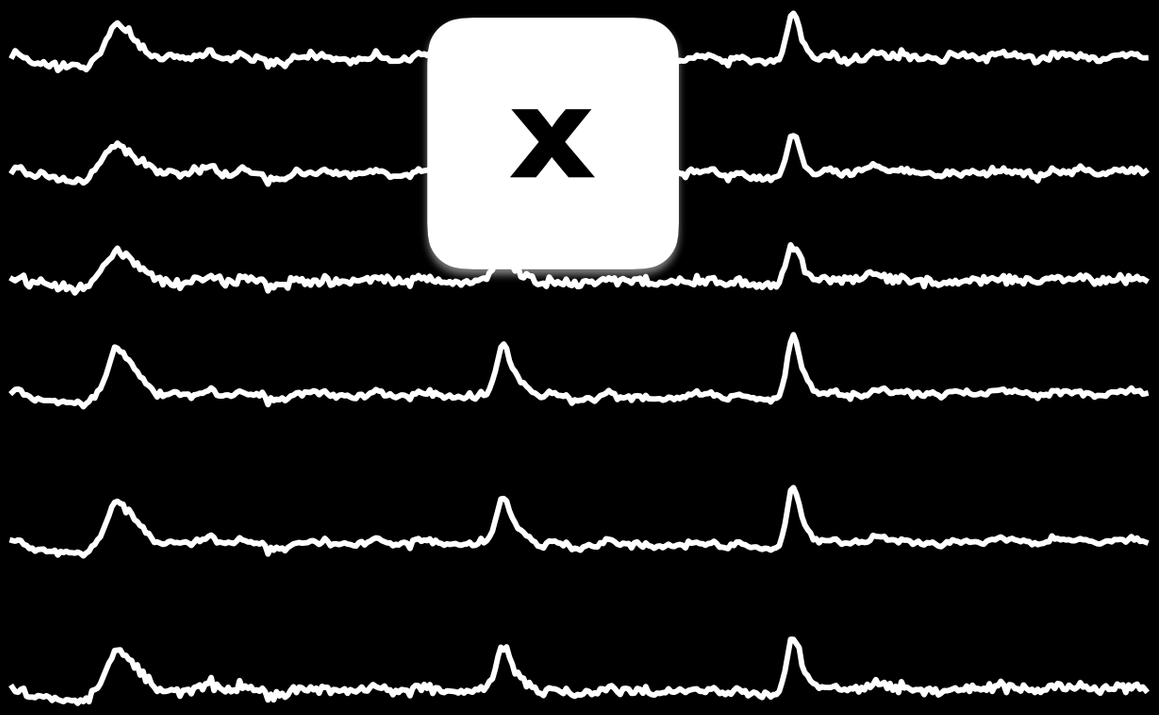


From ICA to CSC

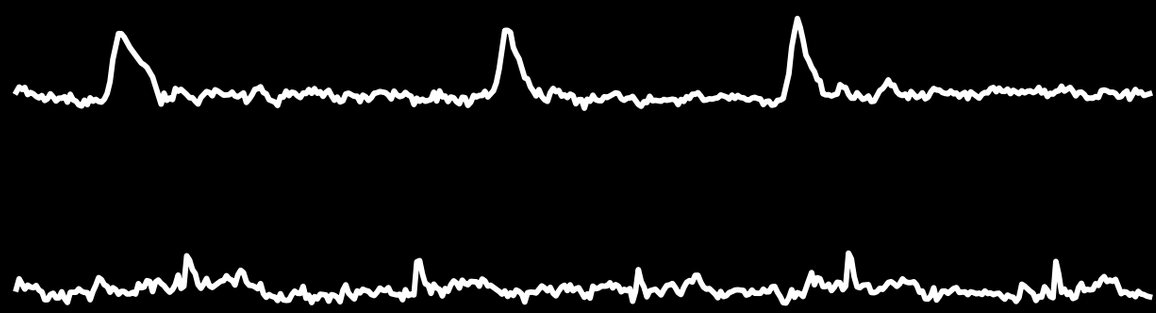
Independent Component Analysis (ICA)

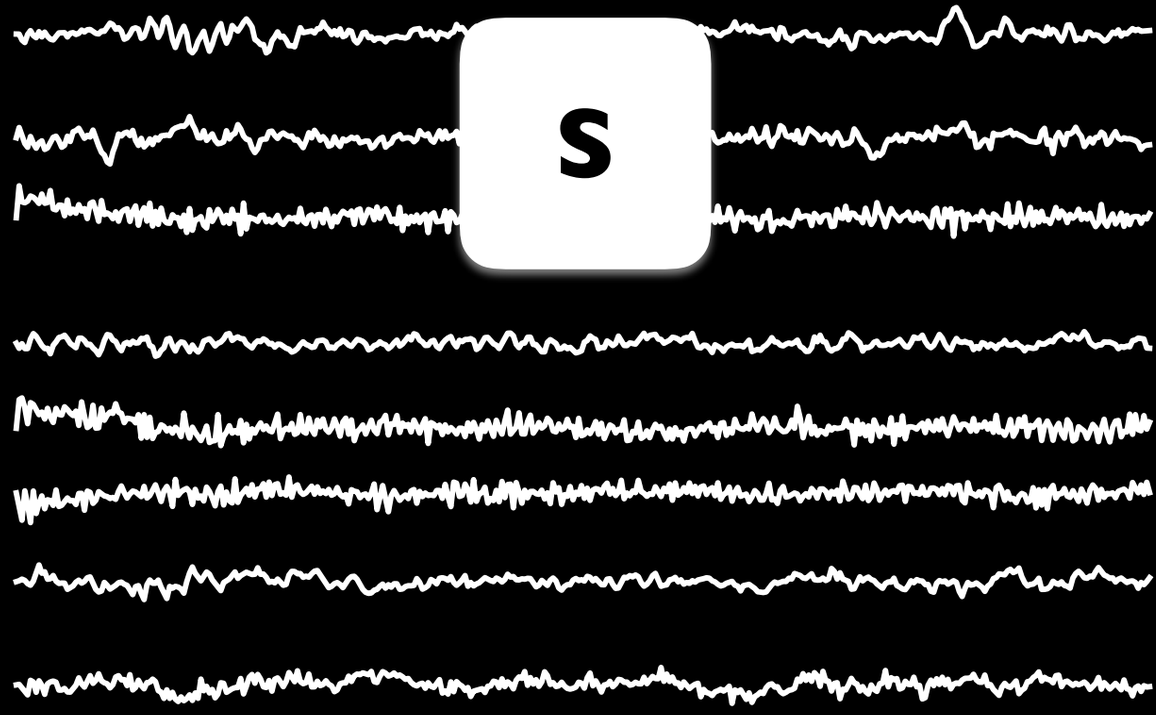
Observations (raw EEG)



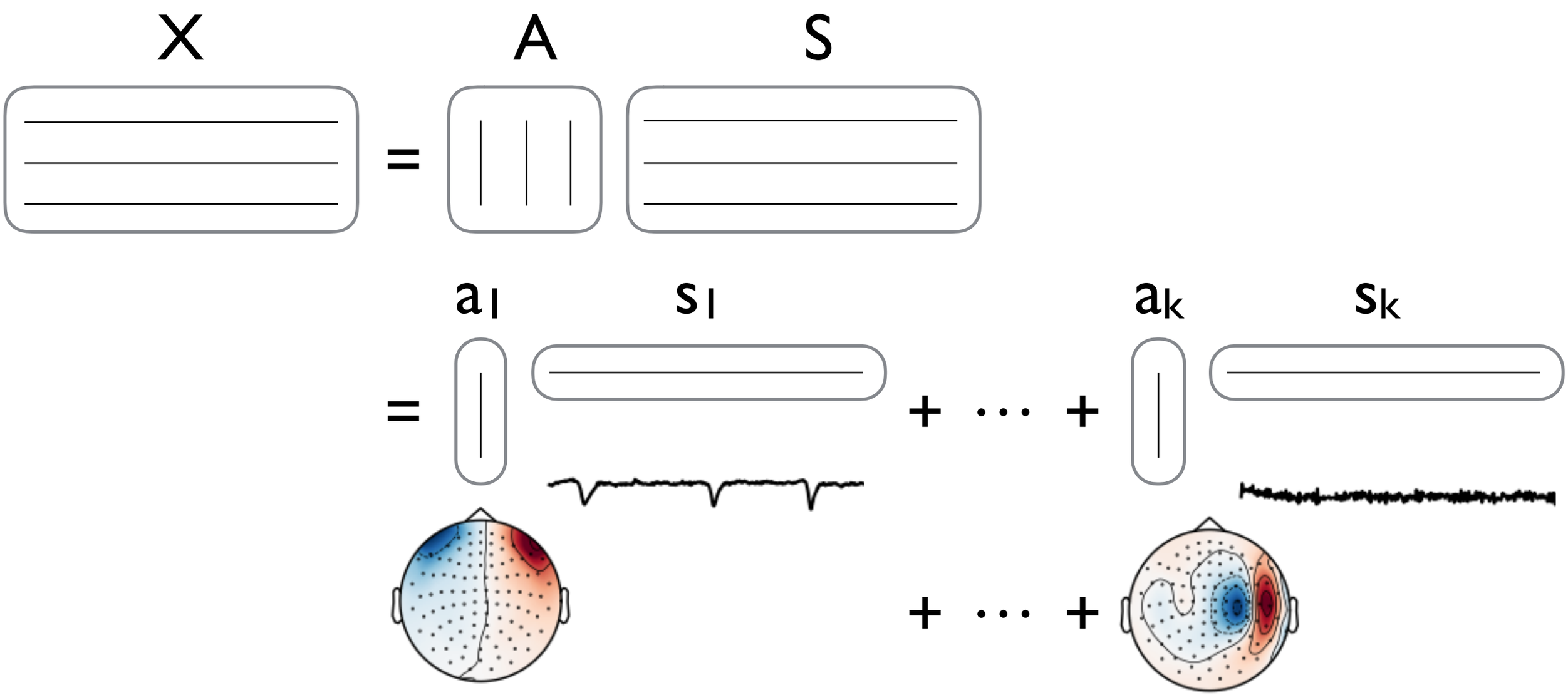


ICA recovered sources





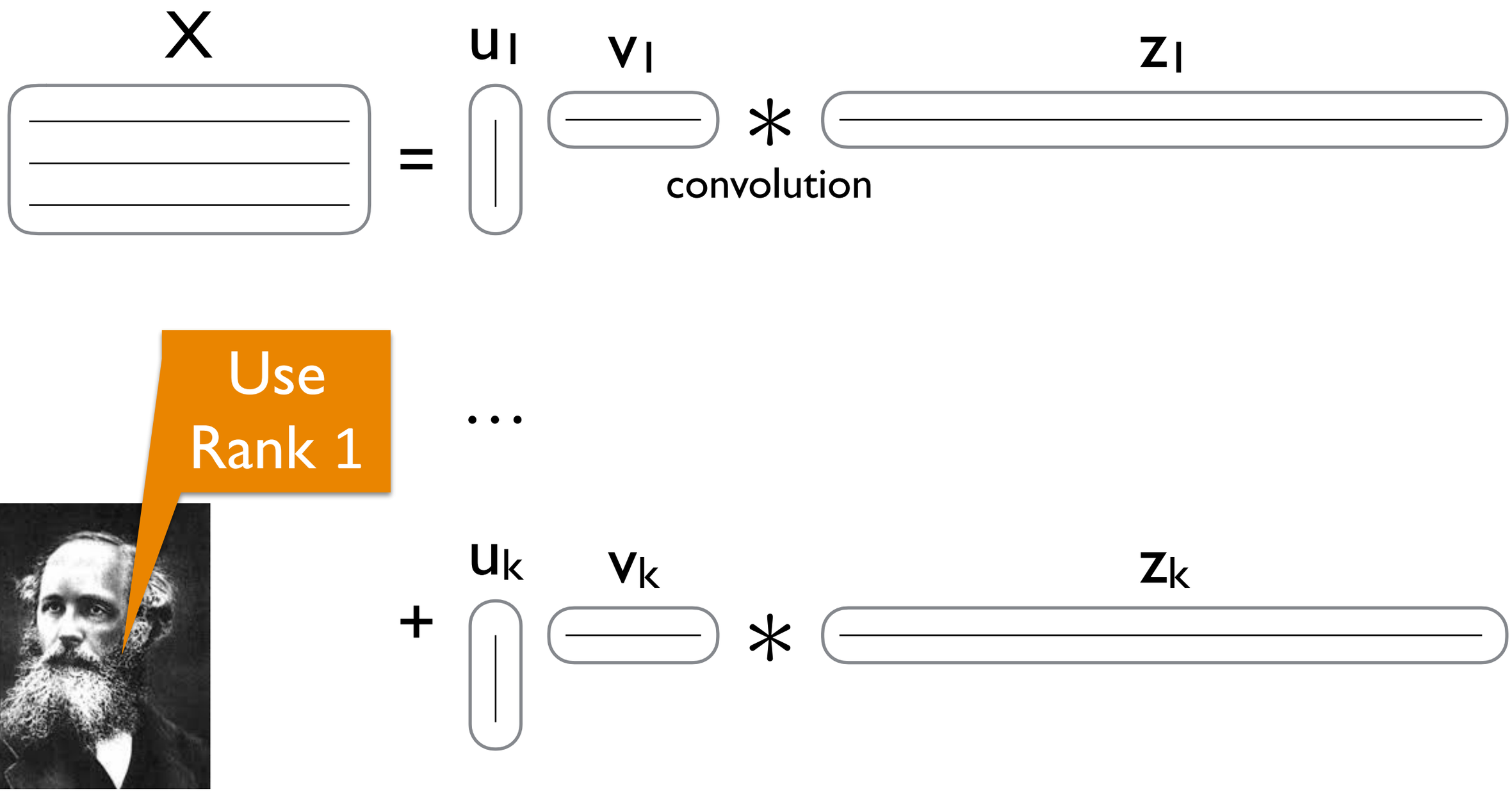
From ICA…



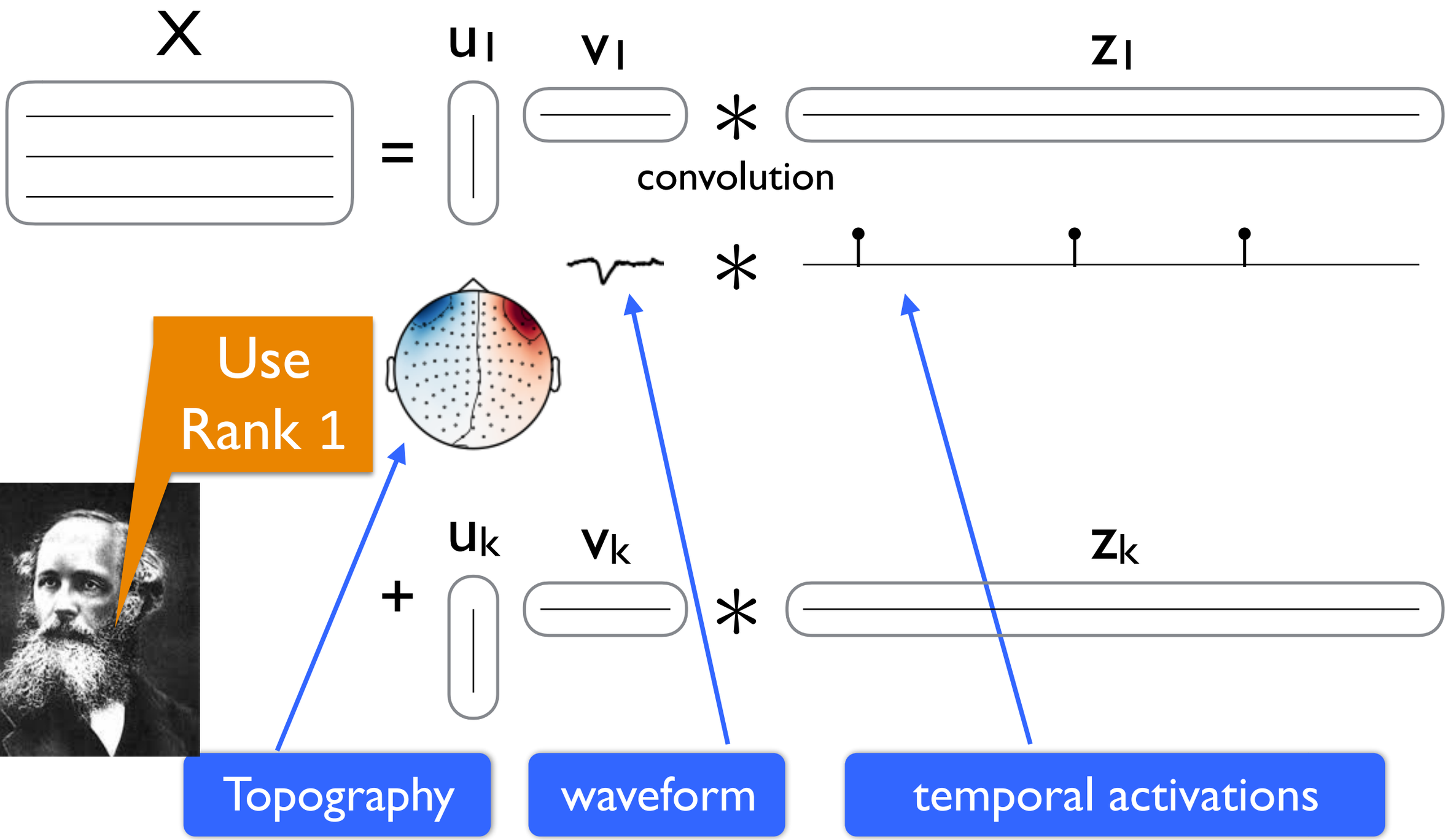
https://www.martinos.org/mne/stable/auto_tutorials/plot_artifacts_correction_ica.html

https://pierreablin.github.io/picard/auto_examples/plot_ica_eeg.html

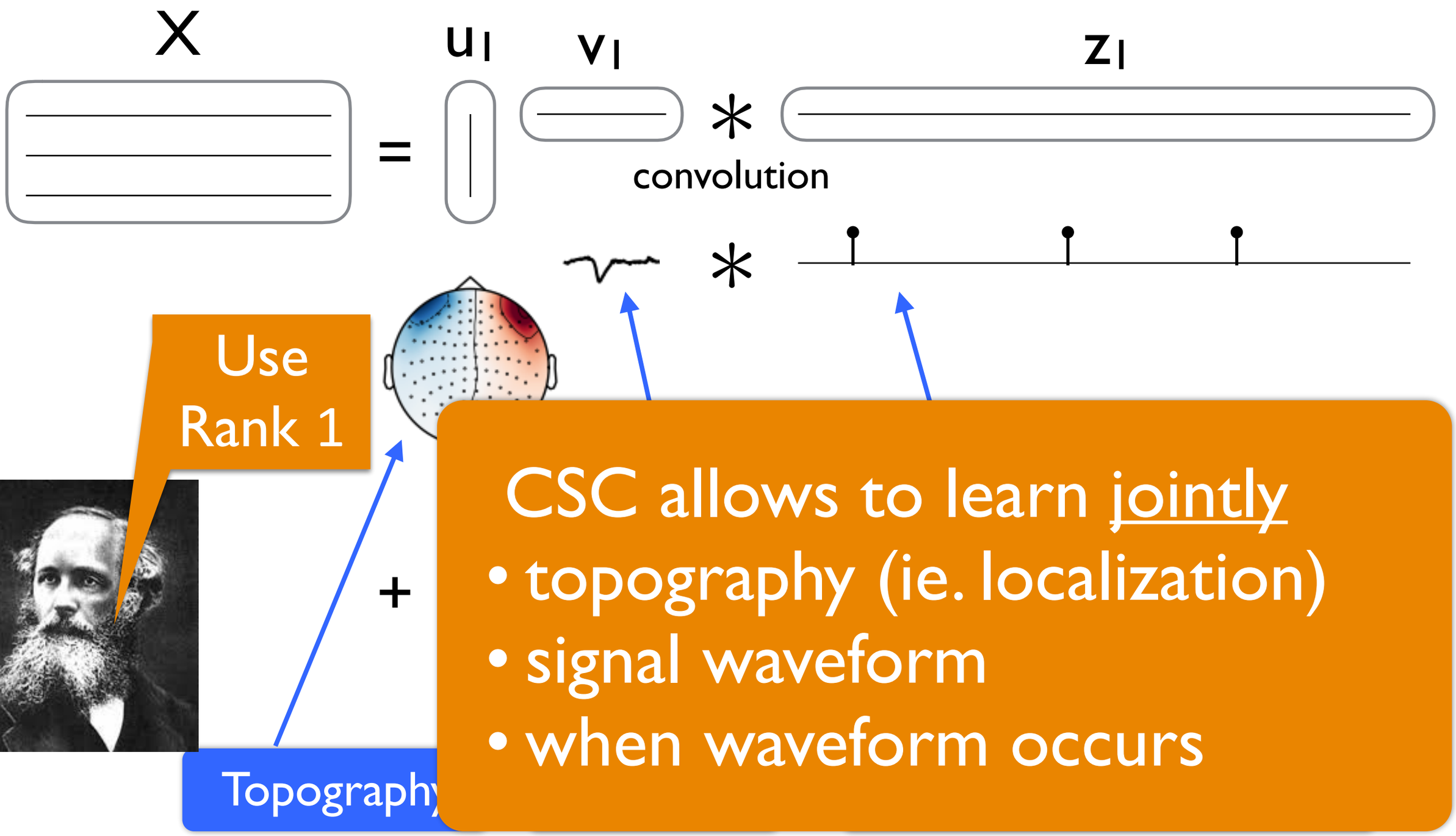
. . . to CSC



… to CSC


X
u1
v1
z1
=
*
convolution
*
Use
Rank 1
uk
vk
zk
+
*
Topography
waveform
temporal activations

… to CSC


X
u_l
v_l
z_l
=
*
convolution
*
Use Rank 1
+
Topography
CSC allows to learn jointly
• topography (ie. localization)
• signal waveform
• when waveform occurs

Multivariate CSC

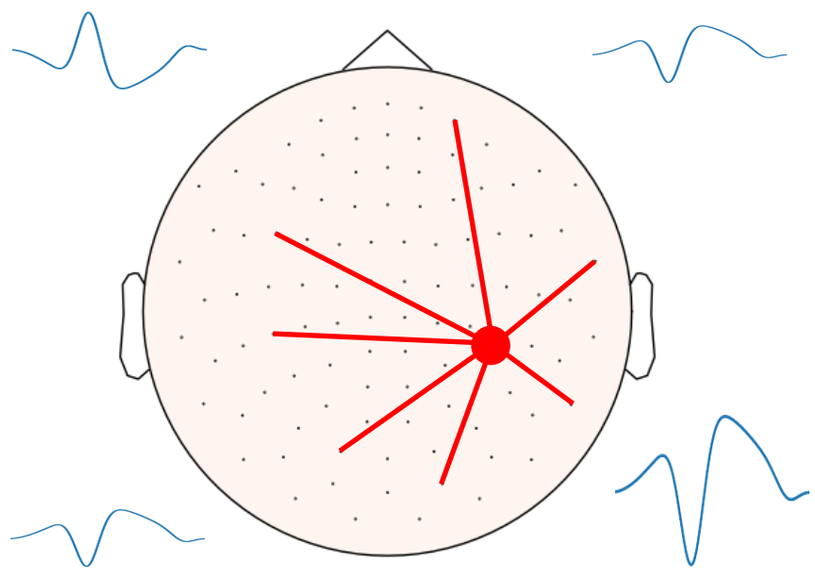
$$\min_{D,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|D_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

Multivariate CSC

$$\min_{D,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$

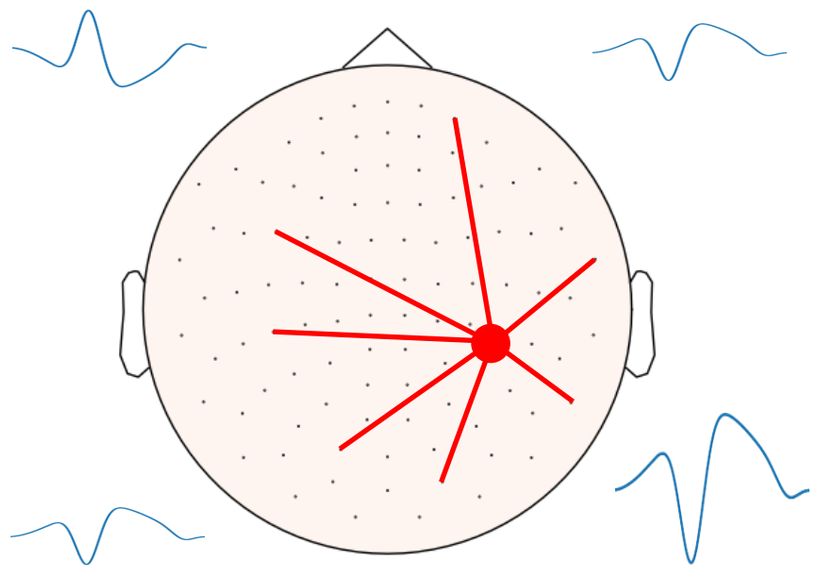
$$\text{s.t.} \quad \|D_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$



[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

Multivariate CSC

$$\min_{D,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|D_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

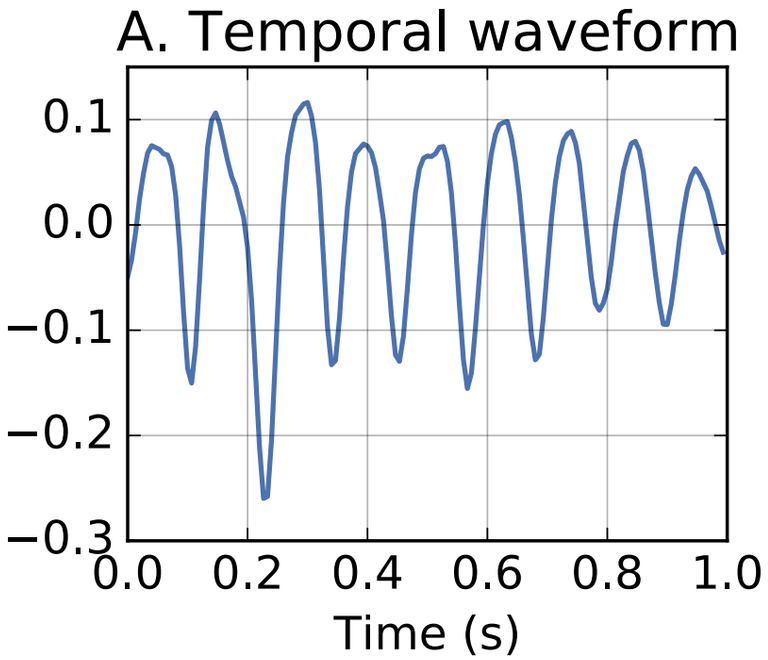


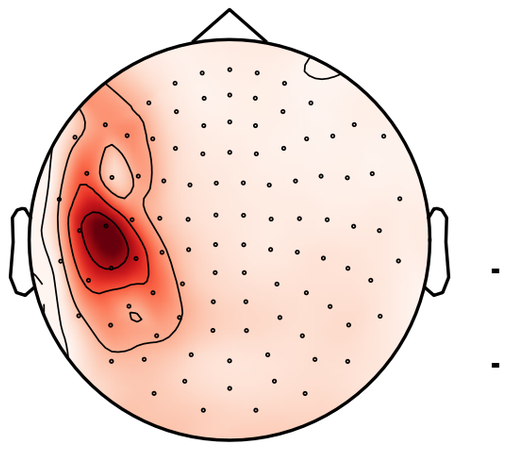
Rank 1 constraint:

$$\min_{u,v,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * (u_k v_k^\top) \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1 ,$$
$$\text{s.t.} \quad \|u_k\|_2^2 \leq 1 \ , \ \|v_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

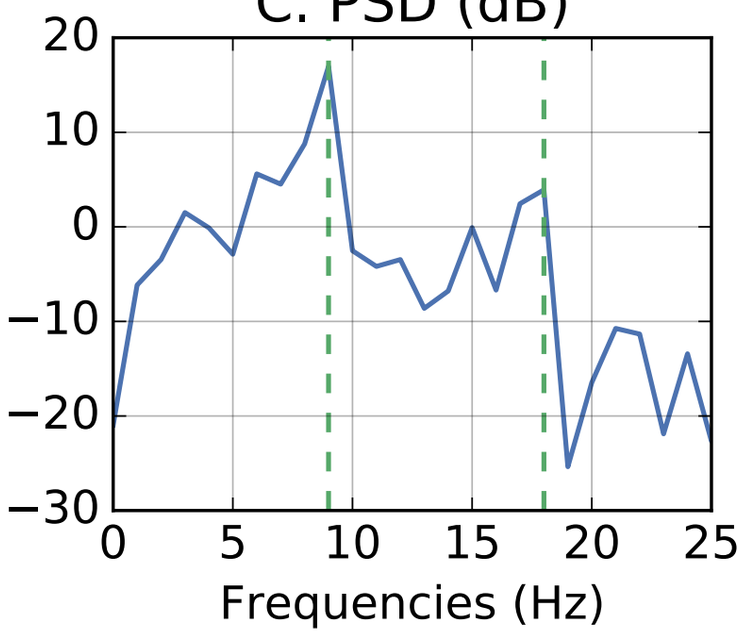
[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

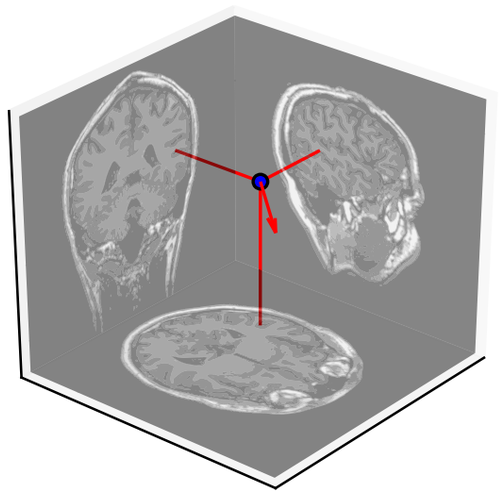
CSC on MEG





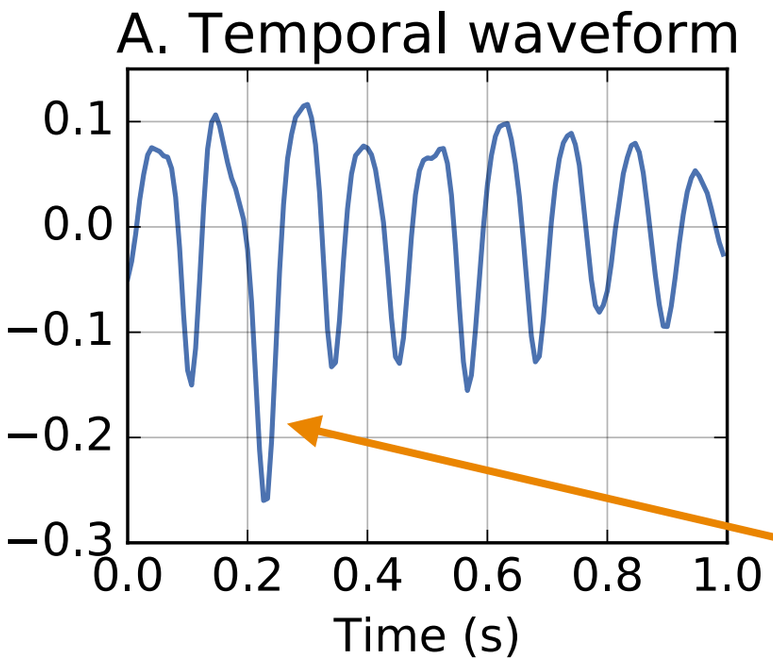
- MEG vectorview
- Median nerve stim.

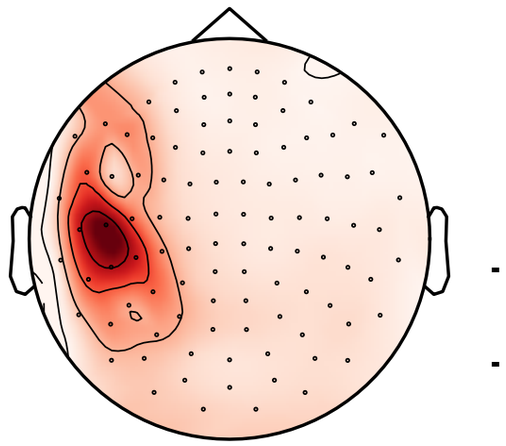




[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

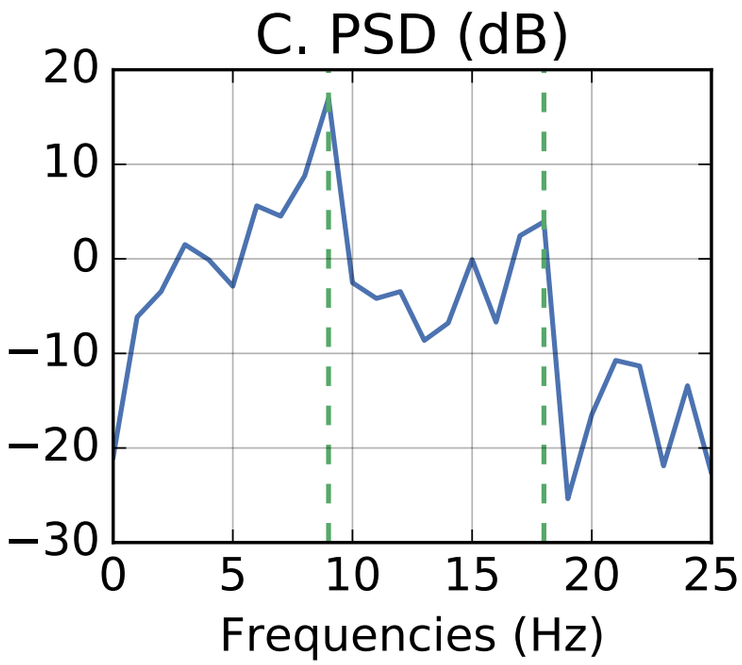
CSC on MEG

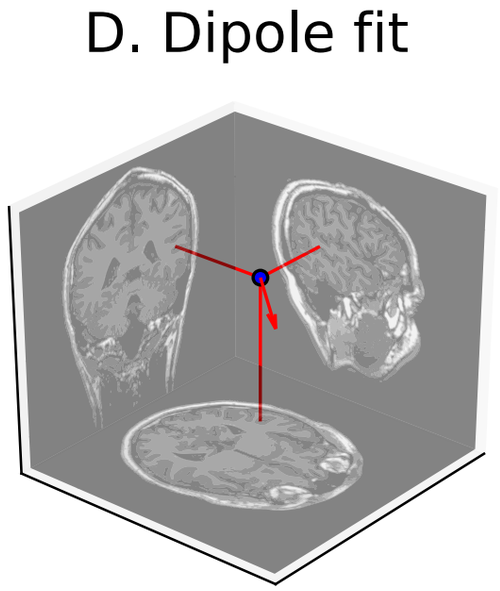




- MEG vectorview
- Median nerve stim.

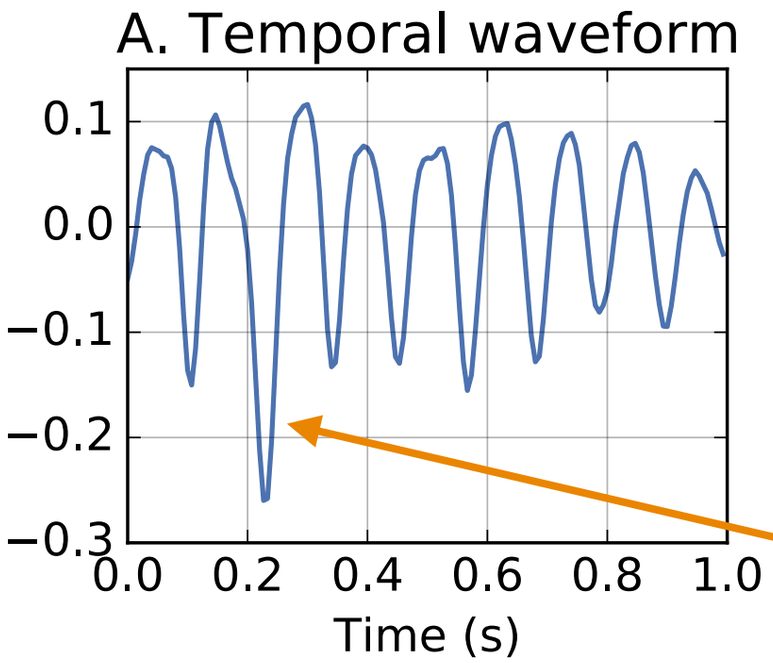
CSC reveals mu-shaped waveforms

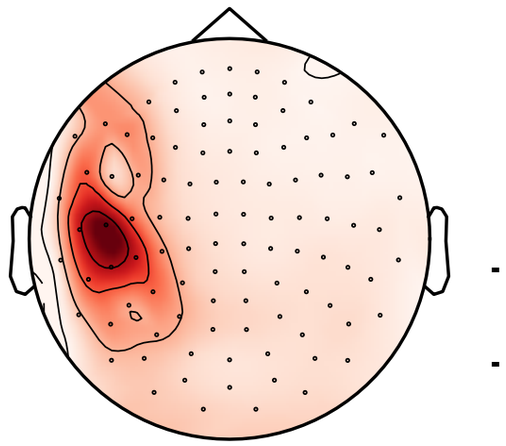




[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

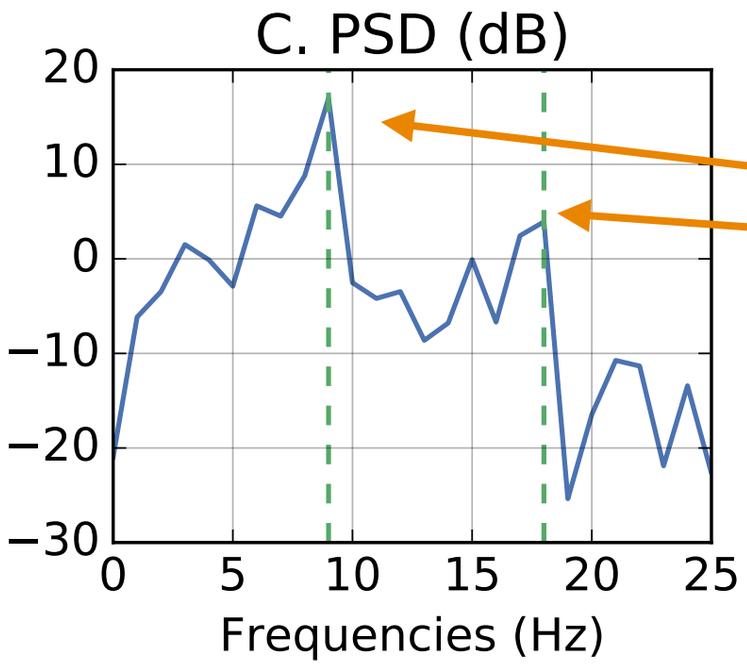
CSC on MEG

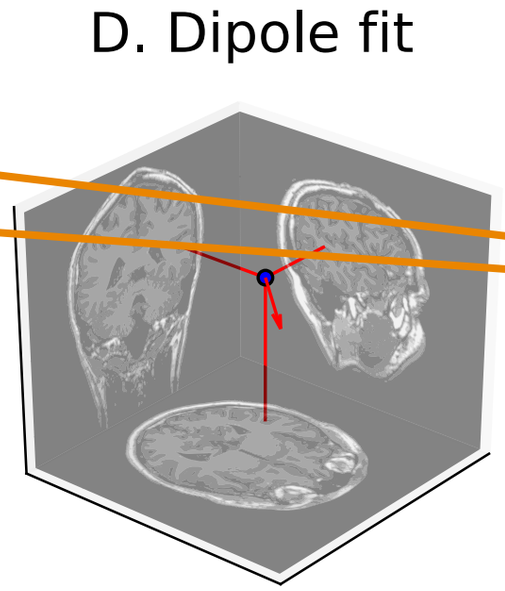




- MEG vectorview
- Median nerve stim.

CSC reveals mu-shaped waveforms

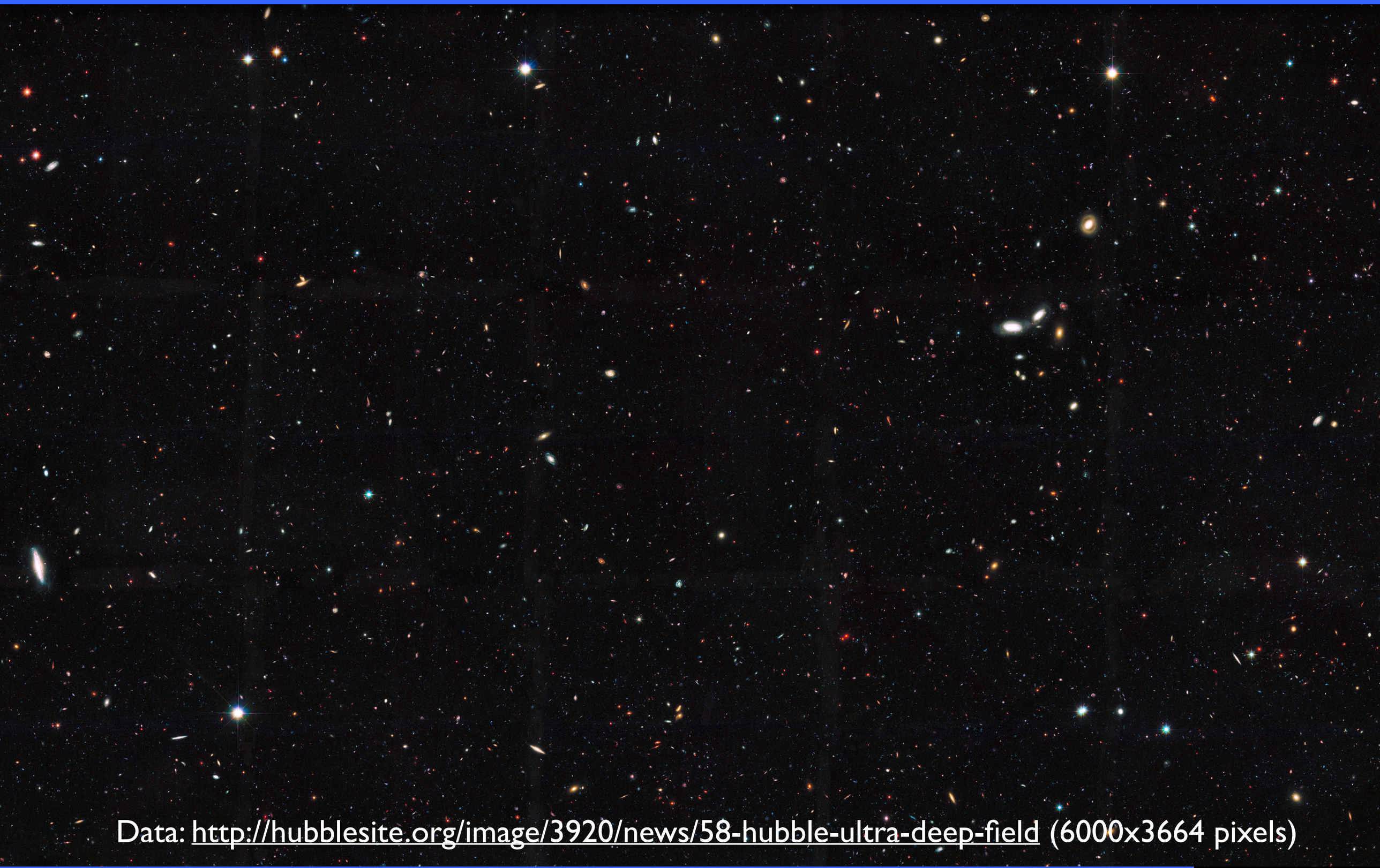




See the frequency harmonics

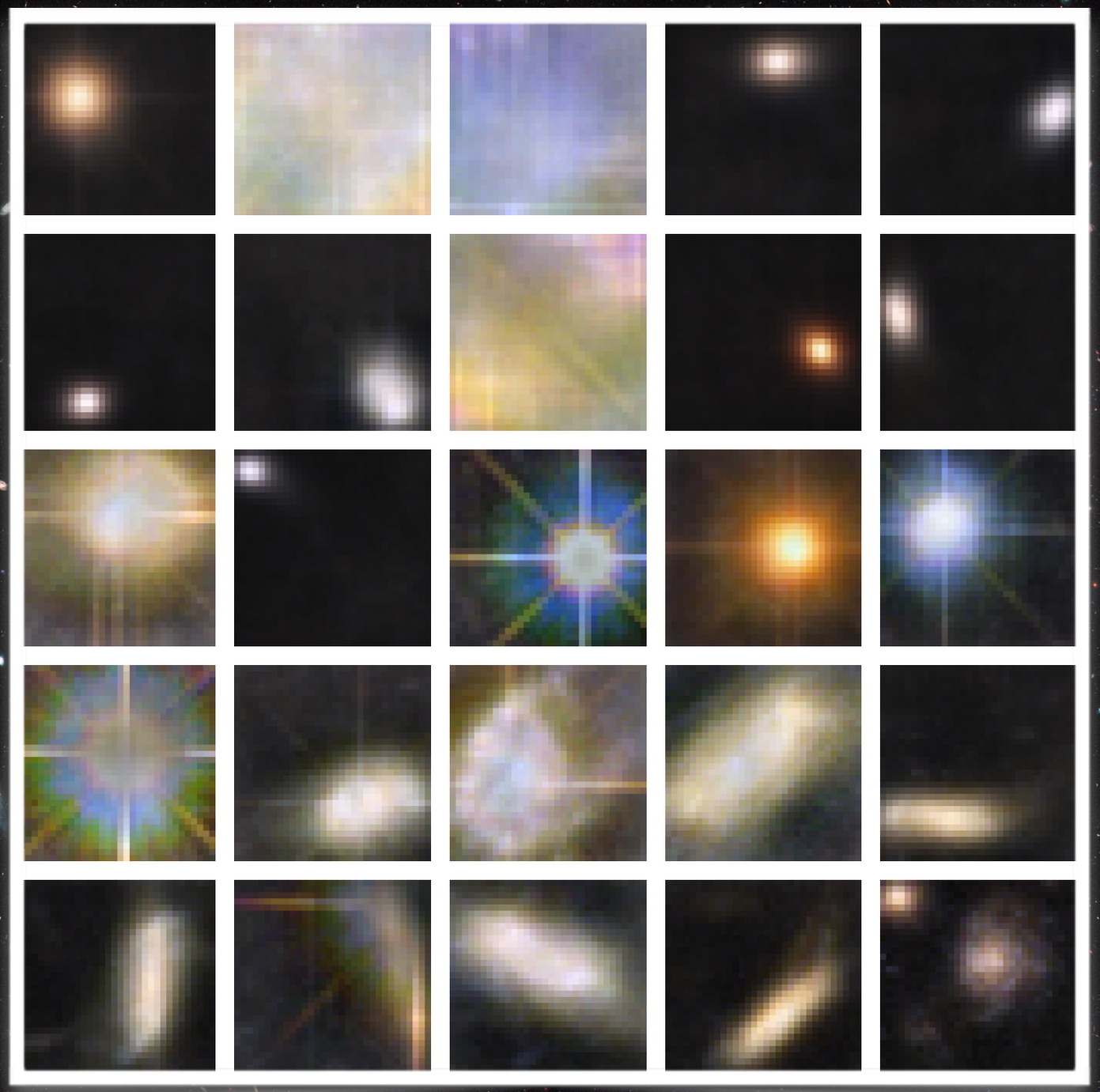
[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

Example using astrophysics



Data: http://hubblesite.org/image/3920/news/58-hubble-ultra-deep-field (6000x3664 pixels)

Example using astrophysics



[Distributed Convolutional Dictionary Learning (DiCoDiLe): Pattern Discovery in Large Images and Signals (2019), T. Moreau and A. Gramfort, ArXiv https://arxiv.org/abs/1901.09235]

Data: http://hubblesite.org/image/3920/news/58-hubble-ultra-deep-field (6000x3664 pixels)

Let's take a step back…

Let's take a step back…

Conclusion

- Contrib 1: Use Hessian approximations to inform a quasi-Newton solver (L-BFGS)
- Contrib 2: Convolutional sparse coding model for multivariate series with fast solvers

Conclusion

- Contrib 1: Use Hessian approximations to inform a quasi-Newton solver (L-BFGS)
- Contrib 2: Convolutional sparse coding model for multivariate series with fast solvers

- Statistical machine learning for neuroscience data
- High dimensional inference with complex noise and non-stationarities
- Fast algorithms for large scale non-smooth problems
- Software tools

"All models are wrong but some come with good open source implementation and good documentation so use those."

Picard

This is a library to run the Preconditioned ICA for Real Data (PICARD) algorithm [1] and its orthogonal version (PICARD-O) [2]. These algorithms show fast convergence even on real data for which sources independence do not perfectly hold.

Installation

We recommend the Anaconda Python distribution. Otherwise, to install `picard`, you first need to install its dependencies:

```
$ pip install numpy matplotlib numexpr scipy
```

Then install Picard:

```
$ pip install python-picard
```

If you do not have admin privileges on the computer, use the `--user` flag with *pip*. To upgrade, use the `--upgrade` flag provided by *pip*.

To check if everything worked fine, you can do:

```
$ python -c 'import picard'
```

and it should not give any error message.

https://pierreablin.github.io/picard/

Comparison of Picard-O and FastICA on faces data

This example compares FastICA and Picard-O:

Pierre Ablin, Jean-François Cardoso, Alexandre Gramfort "Faster ICA under orthogonal constraint" ICASSP, 2018 https://arxiv.org/abs/1711.10873

On the figure, the number above each bar corresponds to the final gradient norm.

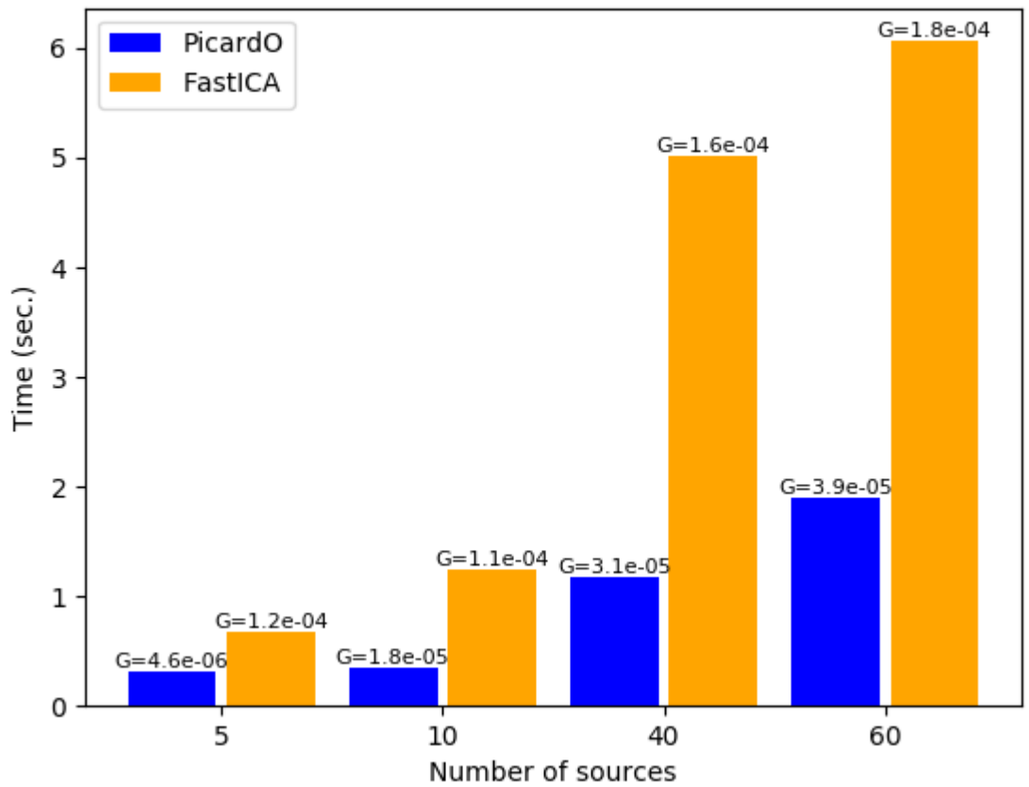
```
# Author: Pierre Ablin <pierre.ablin@inria.fr>
#         Alexandre Gramfort <alexandre.gramfort@inri
# License: BSD 3 clause


import numpy as np
from time import time
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
from sklearn.decomposition import fastica

from picard import picard


print(__doc__)

image_shape = (64, 64)
rng = np.random.RandomState(0)
```



https://pierreablin.github.io/picard/

alphaCSC: Convolution sparse coding for time-series

This is a library to perform shift-invariant sparse dictionary learning, also known as convolutional sparse coding (CSC), on time-series data. It includes a number of different models:

1. univariate CSC
2. multivariate CSC
3. multivariate CSC with a rank-1 constraint [1]
4. univariate CSC with an alpha-stable distribution [2]

A mathematical descriptions of these models is available in the documentation.

Installation

To install this package, the easiest way is using `pip`. It will install this package and its dependencies. The `setup.py` depends on `numpy` and `cython` for the installation so it is advised to install them beforehand. To install this package, please run

```
pip install numpy cython
pip install git+https://github.com/alphacsc/alphacsc.git#egg=alphacsc
```

If you do not have admin privileges on the computer, use the `--user` flag with `pip`. To upgrade, use the `--upgrade` flag provided by `pip`.

To check if everything worked fine, you can run:

```
python -c 'import alphacsc'
```

and it should not give any error messages.

Quickstart

https://alphacsc.github.io

Here is an example to present briefly the API:

```
import numpy as np
```

Thanks !

Especially to:
Mainak Jas (PhD 2014-2017)
Tom Dupré La Tour (PhD 2015-2018)
Pierre Ablin (PhD 2016-…)
Thomas Moreau (Post-doc)
Francis Bach (DR, Inria)
Umut Şimşekli (MdC, Télécom ParisTech)
Jean-François Cardoso (DR, CNRS)

Contact

http://alexandre.gramfort.net

GitHub : @agramfort *Twitter : @agramfort*

Support

ANR-NSF Grant Thalameeg
European Research Council (ERC SLAB-YStG-676943)