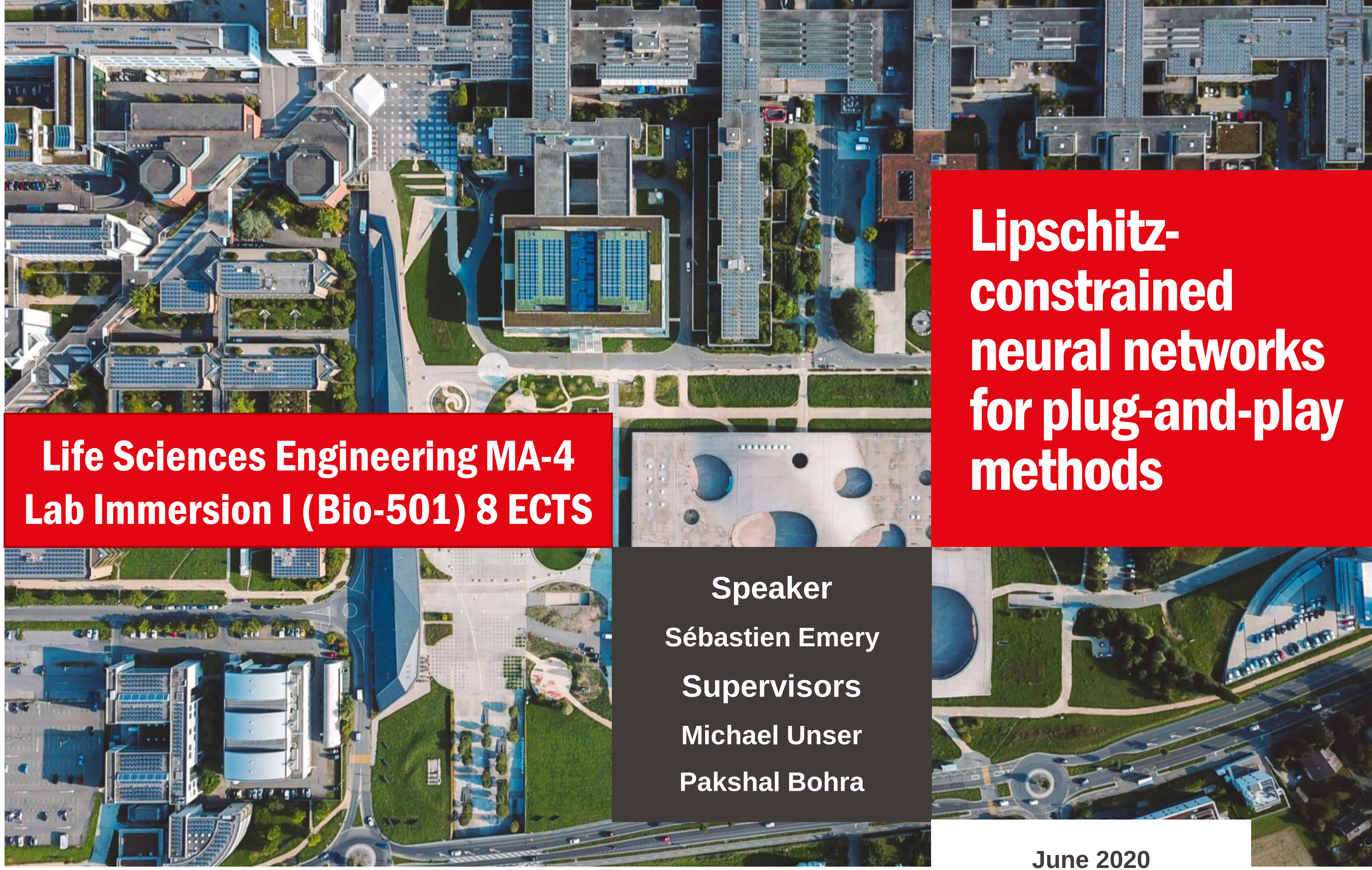
EPFL



Lipschitz-constrained neural networks for plug-and-play methods

Life Sciences Engineering MA-4
Lab Immersion I (Bio-501) 8 ECTS

Speaker

Sébastien Emery

Supervisors

Michael Unser

Pakshal Bohra

June 2020

École polytechnique fédérale de Lausanne

Overview

Introduction

Inverse problems and variational methods

Inverse problems

- **Goal :** Recover a signal through a set of measurements
- **Model :** $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}$ with $\mathbf{y} \in \mathbb{R}^M$, forward operator $\mathbf{H} \in \mathbb{R}^{m x n}$ and noise $\mathbf{n} \in \mathbb{R}^m$
- **Signal to recover** : $\mathbf{x} \in \mathbb{R}^N$

Variational methods

- **Goal :** Find a well-defined optimal solution for inverse problems
- **Optimization problem :** $\hat{\mathbf{x}} = \underset{x}{\operatorname{argmin}} f(\mathbf{x}) + \gamma g(\mathbf{x})$, $\gamma \geq 0$ a constant
  - Data-fidelity term $f(x)$: $f(x) = ||y - \mathrm{H}x||_2^2$ (MSE example)
  - Regularization/Prior term $g(x)$: $g(x) = ||Lx||_1$ (L1-norm example)
- **Algorithms :** ADMM, forward-backward splitting (FBS), …

Plug-and-Play ADMM/FBS

Proximal operator : $\mathrm{Prox}_{\alpha\mathrm{h}}(\mathrm{z}) = \underset{x}{\mathrm{argmin}}\{\alpha \mathrm{h}(x) + (1/2)||x - z||_2^2\}$ with $\alpha > 0$ a constant

ADMM

$$\begin{aligned} \mathrm{x}^{k+1} &= \mathrm{Prox}_{\sigma^2\mathrm{g}}(y^k - u^k) \\ y^{k+1} &= \mathrm{Prox}_{\alpha\mathrm{f}}(x^{k+1} + u^k) \\ u^{k+1} &= u^k + x^{k+1} - y^{k+1} \end{aligned}$$

$\longrightarrow$

Plug-and-Play ADMM

$$\begin{aligned} x^{k+1} &= \mathrm{H}_\sigma(y^k - u^k) \\ y^{k+1} &= \mathrm{Prox}_{\alpha\mathrm{f}}(x^{k+1} + u^k) \\ u^{k+1} &= u^k + x^{k+1} - y^{k+1} \end{aligned}$$

FBS

$$x^{k+1} = \mathrm{Prox}_{\sigma^2\mathrm{g}}(I - \alpha\nabla f)(x^k)$$

$\longrightarrow$

PnP-FBS

$$x^{k+1} = \mathrm{H}_\sigma(I - \alpha\nabla f)(x^k)$$

Idea : Replace a proximal step by a denoisers H_σ with strength parameter $\sigma > 0$

Denoisers : State-of-the-art denoising convolutional networks (CNN), BM3D, …

Plug-and-Play fixed point convergence

Previous work

(Ryu et al. May 2019)[1]

Residual mapping

$$H_\sigma = I - R_\sigma$$

Assumptions

$f(x)$ strongly convex

R_σ is a non-expansive CNN

Current work

(Terris et al. May 2020)[2]

Half-averaged mapping

$$H_\sigma = \frac{I + R_\sigma}{2}$$

Assumptions

None

R_σ is a non-expansive CNN

Non-expansive operator

Operator R: $\mathcal{H} \rightarrow \mathcal{H}$ is non-expansive such that $\forall (x, y) \in \mathcal{H}^2$: $\|Rx - Ry\| \leq \|x - y\|$

α-averaged operator

Operator $T = (1 - \alpha)I + \alpha R$ with $\alpha \in [0, 1]$

Half-averaged: $\alpha = 0{,}5$

1: Ernest K. Ryu et al., Plug-and-play methods provably converge with properly trained denoiser, 2019
2: M. Terris, A. Repetti, J. -C. Pesquet and Y. Wiaux, "Building Firmly Nonexpansive Convolutional Neural Networks," *ICASSP 2020*

Emery Sébastien

CNN architectures

Simple CNN

4 convolutional layers

ReLu or B-splines

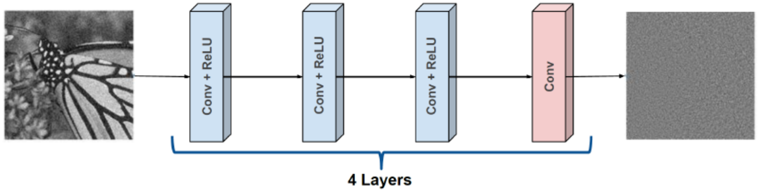
No Batch norm

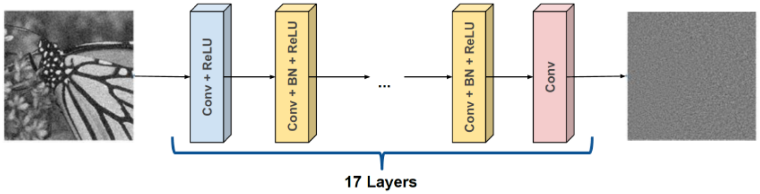
DnCNN

17 convolutional layers

ReLu or B-splines

Batch norm





Train to remove additive white Gaussian noise (AWGN) with variance σ

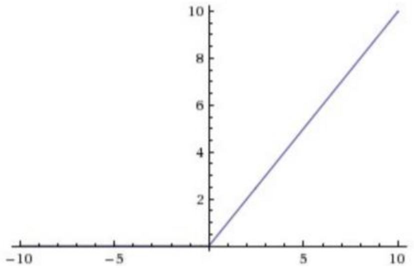
Images taken from : Ernest K. Ryu et al., Plug-and-play methods provably converge with properly trained denoiser, 2019

Lipschitz-constrained ReLu CNNs

Lipschitz-constants

Lipschitz-constant with ℓ_2-norm

- **Definition :** The smallest L satisfying $||f(x) - f(y)||_2 \leqslant L||x - y||_2 \quad \forall x, y \in \mathbb{R}^n$ with $f : \mathbb{R}^n \to \mathbb{R}^m$
- **Composition : L** $\leq \prod_{i=1}^{n} L_i$ for $g = f_n \circ \cdots \circ f_1$
- **Affine transformation** : $W(x) = Wx + b$ **, L** $\leq \sigma_1$ (highest singular value) of W $\in \mathbb{R}^{n \times m}$
- **CNNs :** Control layer-wise the L_i of the different layers and activations



ReLu

→ L is the maximum slope

L = 1

Image taken from : https://junxiaoguo.com/rectifier/relu/2019/08/14/blog-post-4.html

Lipschitz-constrained CNN methods

Fully-connected layer weight matrix : W $\in \mathbb{R}^{n\times m}$
Convolutional Kernel : K $\in \mathbb{R}^{c_{out}\times c_{in}\times h\times w}$

Spectral Normalization[1] (SN)	Real Spectral Normalization[1] (RealSN)	Parseval Normalization[2]
W $\in \mathbb{R}^{n\times m}$ Compute σ_1 of W Power iteration	K $\in \mathbb{R}^{c_{out}\times c_{in}\times h\times w}$ Compute σ_1 of K Kernel power iteration	W $\in \mathbb{R}^{n\times m}$ Keep rows of W orthonormal W $\leftarrow (1+\boldsymbol{\beta})W - \boldsymbol{\beta}WW^TW$
<u>Fully-connected layer</u> Constrain W with σ_1 1 iteration	<u>Fully-connected layer</u> -	<u>Fully-connected layer</u> Constrain W
<u>Convolutional layer</u> Reshape K to W W $\in \mathbb{R}^{c_{out}\times(c_{in}\cdot h\cdot w)}$ Constrain W with σ_1 1 iteration	<u>Convolutional layer</u> Constrain K with σ_1 1 iteration	<u>Convolutional layer</u> Reshape K to W W $\in \mathbb{R}^{c_{out}\times(c_{in}\cdot h\cdot w)}$ Divide output by $\sqrt{h}\cdot\sqrt{w}$ Constrain W

1: Ernest K. Ryu et al., Plug-and-play methods provably converge with properly trained denoiser, 2019
2: Moustapha Cisse et al., Parseval Networks: Improving Robustness to Adversarial Examples, 2017

Lipschitz experiments

Experimental set-up

- Architecture: Simple CNN
- NYU fastMRI[1] dataset : Knee MRI images
- Mappings : Residual (R) and Half-averaged (H)
- Noise variance : $\sigma = 0{,}05$

β=0,5

Comparison of the methods
No normalization (None)

		None		SN		RealSN		Parseval	
		R	H	R	H	R	H	R	H
Singular values	Layer 1	4.81	5.9	5.00	5.2	1.03	1.04	3.0	3.0
	Layer 2	19.42	27.14	14.05	12.17	1.01	1.0	2.94	2.98
	Layer 3	19.68	11.91	11.92	9.69	1.04	1.05	2.56	2.98
	Layer 4	1.2	0.63	3.19	3.51	1.01	1.01	1.06	2.96

Parseval experiment
β : strength parameter
Residual mapping

Beta		0.0001	0.1	0.5	0.6	0.7	1.0
Singular values	Layer 1	6.25	3.00	3.00	3.00	NAN	NAN
	Layer 2	4.82	2.0	2.94	2.98	NAN	NAN
	Layer 3	4.24	1.65	2.56	2.96	1.88	1.34
	Layer 4	2.58	1.05	1.06	2.88	1.14	0.99

Singular values estimated by Kernel power iterations (100) on trained networks

1: https://fastmri.med.nyu.edu/

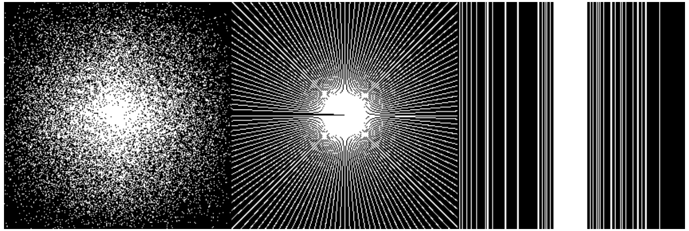
Compressed sensing MRI

Model

- **Measurements :** $\mathrm{y} = \mathcal{F}_p \mathrm{x}_{\text{true}} + \varepsilon_e$
  - **Noise :** $\varepsilon_e \sim \mathcal{N}(0, \sigma_e I_m)$, $\sigma_e = 15/255$
  - **Forward operator :** $\mathcal{F}_p : \mathbb{C}^n \to \mathbb{C}^m$ Fourier K-space subsampling
- **Data-fidelity term :** $f(x) = \frac{1}{2}||\mathrm{y} - \mathcal{F}_p \mathrm{x}||_2^2$ not strongly convex
- **Regularization term :** CNN based denoisers H_σ

Sampling patterns

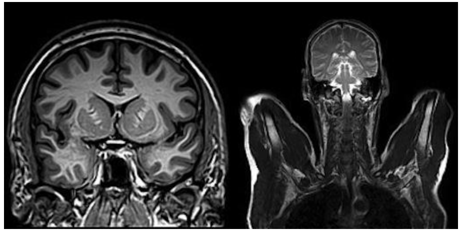
Random Radial Cartesian



- Sampling rate : 30%

Images

Brain Bust



PnP methods

CNNs

- Architectures : SimpleCNN (4 layers) or DnCNN (17 layers with batch norm)
- Mappings : Residual (R) or Half-averaged (H)
- Spectral Normalization : RealSN and None
- Activations : ReLu
- Noise Level σ : 15/255 or 5/255
- Convolutional layers : With or without bias

→ Train 32 network combinations on **natural images** (BSD500[1])

PnP

- **ADMM :** Networks with σ = 15/255 were plugged-in (20) with $\alpha = 2.0$
- **FBS :** Networks with σ = 5/255 were plugged-in (20) with $\alpha = 0.4$
- All combinations (6) of images and sampling tested for 100 iterations

Baseline

- Inverse 2-D Fourier Transform (zero-filling)

1: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

PnP results

Networks with bias

Sampling approach		Random				Radial			
Image		Brain		Bust		Brain		Bust	
CNN architecture		R	H	R	H	R	H	R	H
Zero-filling		9.60		7.02		9.29		6.21	
PnP-FBS	DnCNN	**19.84**	**18.87**	**16.82**	15.60	**19.21**	**18.30**	**16.33**	15.07
	Real-SN DnCNN	19.60	18.15	16.49	15.10	18.73	17.41	15.98	14.26
	SimpleCNN	19.21	18.81	16.61	**15.89**	18.42	18.08	16.09	**15.45**
	Real-SN SimpleCNN	18.57	16.91	16.23	13.83	17.74	16.42	15.75	13.37
PnP-ADMM	DnCNN	**19.80**	**19.83**	16.99	**17.09**	**18.85**	**18.91**	**16.61**	**16.61**
	Real-SN DnCNN	19.75	19.10	**17.10**	16.47	18.75	17.98	16.55	15.94
	SimpleCNN	19.49	19.44	16.96	16.95	18.54	18.45	16.44	16.43
	Real-SN SimpleCNN	18.79	16.87	16.48	14.69	17.79	15.96	15.93	13.92

Drop in performance using Half-averaged instead of Residual (0,5 - 2 dB)
Only PnP-ADMM DnCNN is slightly higher

→ Replace ReLu by B-splines to try to reduce the gap

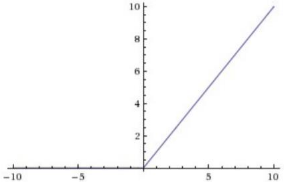
Lipschitz-constrained B-spline CNNs

Constrained B-spline activations

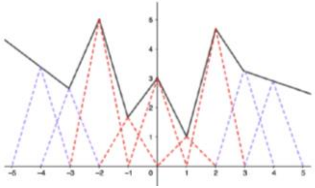
B-spline activations

- Increase capacity
- Uniform grid with step size T (k coefficients)
- Linear extrapolation outside the grid (2 coefficients)
- Convolutions
  - Shared within channel , size : (number channels, (K+2))

ReLu



B-spline



Lipschitz-constrained B-spline activations

- One dimensional function
- Continuous piecewise linear
- L is the maximum slope

Quadratic constrained minimization problem

1. For m in B-spline modules **do**
   2. For n = {1,2,...,number of channels} **do**
      3. $\tilde{c} = c$ where $c \in \mathbb{R}^{K+2}$
      4. Solve $||\bar{c} - \tilde{c}||_2^2$, subject to $D\tilde{c} \geqslant 1$ with D the first order finite difference
      5. c = $\bar{c}$

Python libraries : cvxpy/cvxpylayers and qpth

Maximum Projection

1. For m in B-spline modules **do**
   2. For n = {1,2,...,number of channels} **do**
      3. **Initialize** S list of slopes
      4. For c = {1,2,...,C-1} **do**
         5. Compute absolute slope {c,c+1} abs(s)
         6. Append s to S
      7. $s^* = max(S)$
      8. **If** $s^* \leqslant 1$:
         do nothing
      9. **else** $s^* \geqslant 1$:
         For c = {1,2,...,C-1} **do** :
            c = c/ s^*

B-spline activation image taken from : P. Bohra, J. Campos, H. Gupta, S. Aziznejad, and M. Unser, Learning activation functions in deep (spline) neural networks, 2020

B-spline activations experiments

Computational Cost

- Architectures: RealSN Simple CNN and RealSN DnCNN
- Mapping: Half-averaged
- K = 51 (coefficients) with T = 0.1 (step size)
- Dataset: BSD500
- Noise: $\sigma = 15/255$

Model	SimpleCNN	DnCNN
ReLu	4min40s	7min44s
B-spline	5min49s	12min53s
B-spline maxproj	5min53s	13min7s
B-spline orthoprojection	25min19s	1h55min

→ **Prohibitive**

Time per epoch

PnP methods

CNNs

- Architectures : SimpleCNN (4 layers) or DnCNN (17 layers with batch norm)
- Mappings : **Half-averaged (H)**
- Spectral Normalization : RealSN
- Activations *:* **B-splines** or ReLu
- Noise Level σ : 15/255 or 5/255
- Convolutional layers : With or without bias

→ Train 16 network combinations on **natural images** (BSD500)

PnP

- **ADMM :** Networks with σ = 15/255 were plugged-in (20) with $\alpha = 2.0$
- **FBS :** Networks with σ = 5/255 were plugged-in (20) with $\alpha = 0.4$
- All combinations (6) of images and sampling tested for 100 iterations

Baseline

- Inverse 2-D Fourier Transform (zero-filling)

PnP results (1)

Networks with bias

Sampling approach		Random		Radial	
Image		Brain	Bust	Brain	Bust
CNN architecture		H	H	H	H
Zero-filling		9.60	7.02	9.29	6.21
PnP-FBS	Real-SN DnCNN	18.15	15.10	17.41	14.26
	Real-SN DnCNN Bspline$^+$	14.24	12.98	12.74	11.99
	Real-SN SimpleCNN	16.91	13.83	16.42	13.37
	Real-SN SimpleCNN Bspline	17.55	14.81	16.91	14.34
PnP-ADMM	Real-SN DnCNN	19.10	16.47	17.98	15.94
	Real-SN DnCNN Bspline	19.21	16.83	18.39	16.27
	Real-SN SimpleCNN	16.87	14.69	15.96	13.92
	Real-SN SimpleCNN Bspline	17.11	14.97	16.28	14.28

PnP-FBS

DnCNN : B-spline performance much worse

Simple CNN : B-spline performance better

PnP-ADMM

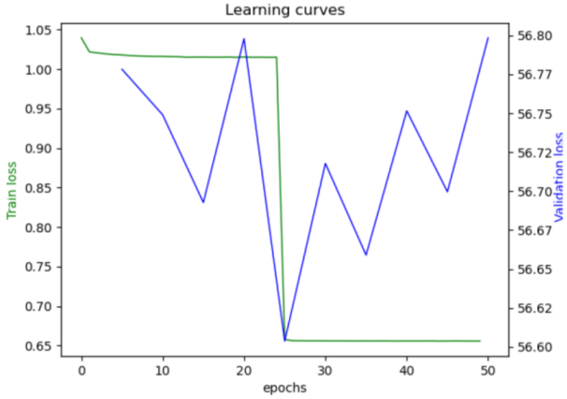
Slight improvement with B-spline

PnP results (2)

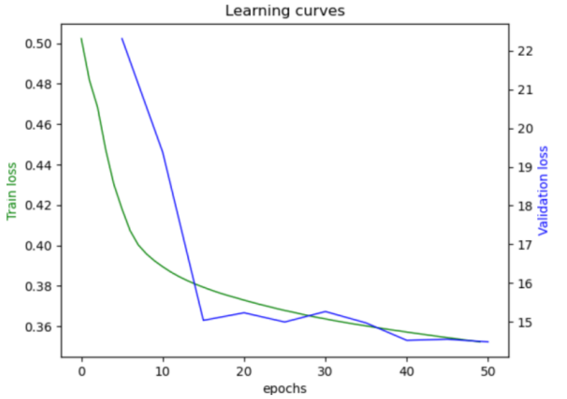
B-spline activations

- Difficult learning process
  - Adam
    - ➢ Validation curve oscillates
    - ➢ Dependent on the initialization
    - ➢ Lower learning rate does not solve the issue
  - SGD
    - ➢ Validation curve decreases
    - ➢ No momentum
    - ➢ Lower learning rate solve the issue
- Default settings
  - No hyperparameter tuning

PnP-FBS RealSN DnCNN B-spline



Adam



SGD

Denoiser Scaling

Denoising strength

Motivation

- Noise measurements might not be available
- Optimal σ tuning

Direct approach

σ : noise added to the images

Changing σ involves retraining a network

Scaling[1] approach

σ : noise added to the images

μ : scaling parameter

$$\longrightarrow \quad D_\mu(z) = \frac{1}{\mu} D(\mu z)\,,\, z \in \mathbb{R}^n$$

D : trained denoiser where σ is dropped

z : noisy input

1: Xiaojian Xu et al., Boosting the Performance of Plug-and-Play Priors via Denoiser Scaling, 2020

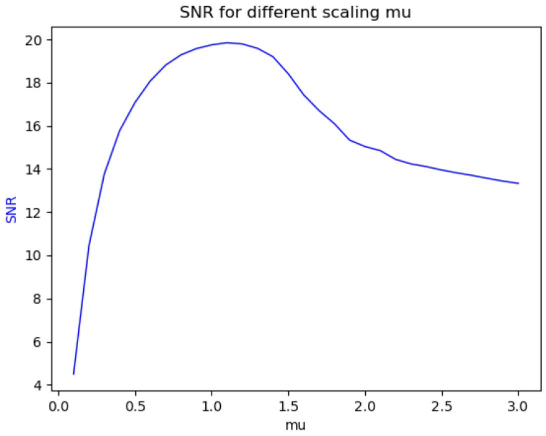
Scaling experiments

Values of noises

- Measurement's noise : $\sigma_e = 15/255$
- PnP-ADMM denoisers : $\sigma = 15/255$
- PnP-FBS denoisers : $\sigma = 5/255$

Scaling

- Prior knowledge of σ_e
- Scale denoisers train on a *"guessed"* σ
- Scaling parameter μ
  - Uniform grid [0.1-3.0]
  - Step size T = 0.1



$\boxed{\mu^* = 1.1}$

PnP-ADMM : Residual RealSN DnCNN with Random sampling

Future directions

- Test the PnP framework on other modalities
  - Case of interest : Residual mapping does not converge
- Investigate B-spline activations
  - Careful investigation of the learning process
  - Tuning hyperparameters
- Testing the scaling method
  - No prior knowledge of measurement’s noise σ_e

EPFL

Thank you

Emery Sébastien

Implementation in Python

https://github.com/sebemery/Lipschitz-constrained-neural-networks

Appendix

Complete PnP result table for network with bias

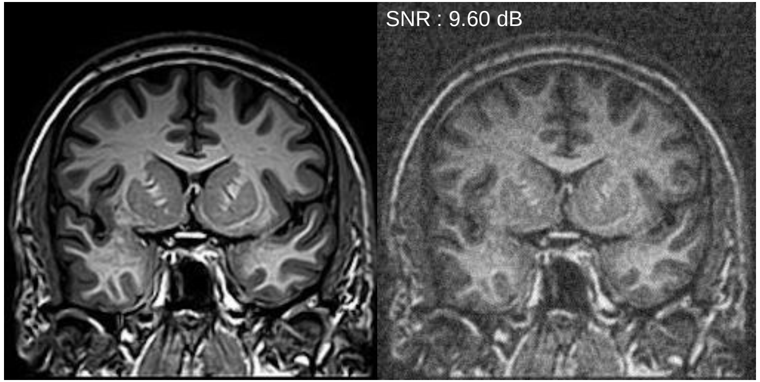
Sampling approach		Random				Radial				Cartesian			
Image		Brain		Bust		Brain		Bust		Brain		Bust	
CNN architecture		R	H	R	H	R	H	R	H	R	H	R	H
Zero-filling		9.60		7.02		9.29		6.21		8.67		6.03	
PnP-FBS	DnCNN	**19.84**	**18.87**	**16.82**	15.60	**19.21**	**18.30**	**16.33**	15.07	**15.00**	14.27	14.22	13.43
	Real-SN DnCNN	19.60	18.15	16.49	15.10	18.73	17.41	15.98	14.26	14.19	13.70	13.54	12.87
	Real-SN DnCNN Bspline^{+}	17.35	14.24	15.48	12.98	16.81	12.74	14.60	11.99	13.55	9.51	12.22	9.20
	SimpleCNN	19.21	18.81	16.61	**15.89**	18.42	18.08	16.09	**15.45**	14.82	**14.55**	**14.30**	**13.72**
	Real-SN SimpleCNN	18.57	16.91	16.23	13.83	17.74	16.42	15.75	13.37	14.38	12.99	13.82	11.46
	Real-SN SimpleCNN Bspline	19.05	17.55	16.57	14.81	18.13	16.91	16.09	14.34	14.80	13.15	13.80	11.92
PnP-ADMM	DnCNN	**19.80**	**19.83**	16.99	**17.09**	**18.85**	**18.91**	**16.61**	**16.61**	**15.07**	**14.70**	**13.99**	13.61
	Real-SN DnCNN	19.75	19.10	**17.10**	16.47	18.75	17.98	16.55	15.94	14.74	14.32	13.75	13.63
	Real-SN DnCNN Bspline	19.14	19.21	16.34	16.83	18.26	18.39	16.18	16.27	14.21	14.49	12.82	13.39
	SimpleCNN	19.49	19.44	16.96	16.95	18.54	18.45	16.44	16.43	14.65	14.67	13.82	**13.82**
	Real-SN SimpleCNN	18.79	16.87	16.48	14.69	17.79	15.96	15.93	13.92	13.83	12.57	12.73	10.63
	Real-SN SimpleCNN Bspline	18.74	17.11	16.52	14.97	17.69	16.28	15.97	14.28	13.80	12.71	12.85	11.00

Appendix

PnP-ADMM

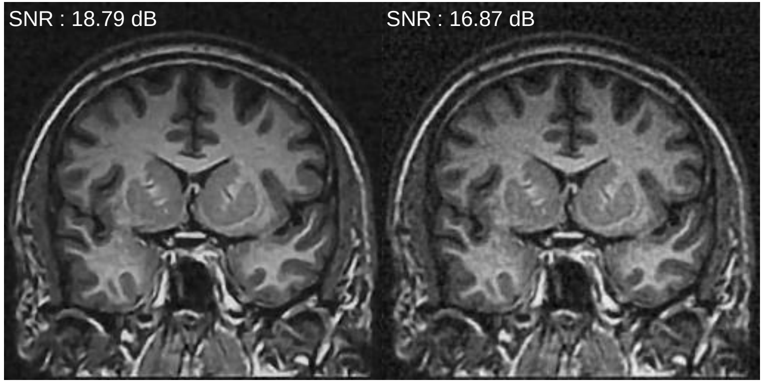
Random sampling

Ground Truth Zero-Filling



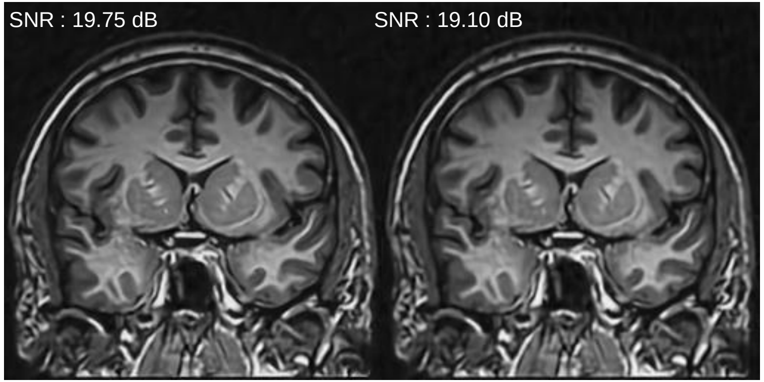
RealSN Simple CNN

Residual Half-averaged



RealSN DnCNN

Residual Half-averaged



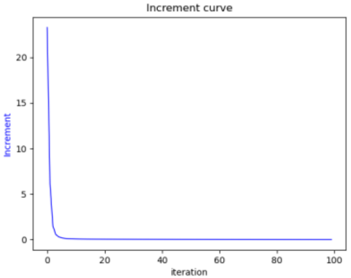
Appendix

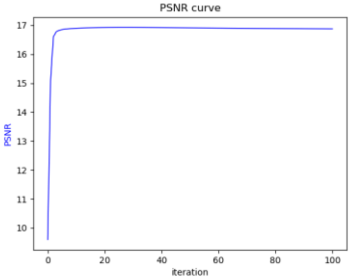
PnP-ADMM
Random sampling

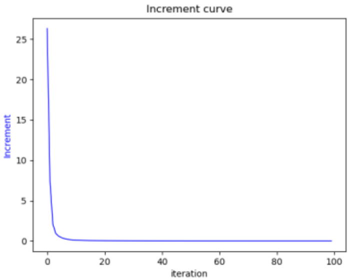
RealSN Simple CNN

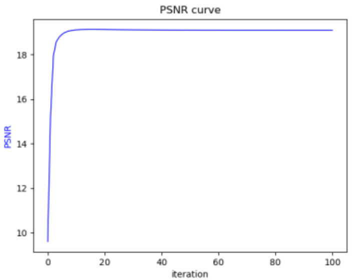
RealSN DnCNN

Half-averaged

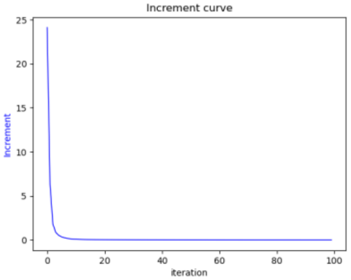


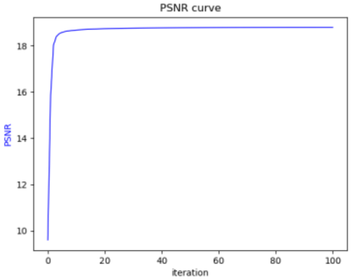


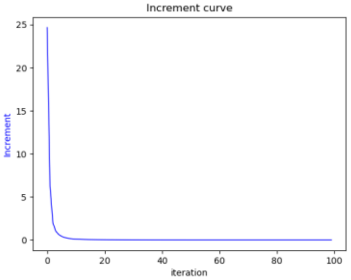


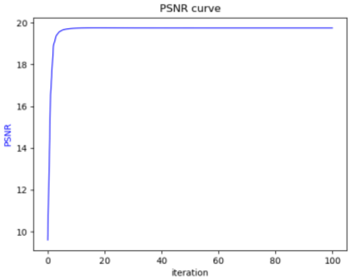


Residual









Appendix

Sampling approach		Random								Radial							
Image		Brain				Bust				Brain				Bust			
CNN architecture		R	R*	H	H*	R	R*	H	H*	R	R*	H	H*	R	R*	H	H*
Zero-filling		9.60				7.02				9.29				6.21			
PnP-FBS	DnCNN	**19.84**	**19.84**	**18.87**	**18.87**	**16.82**	**16.82**	15.60	15.79	**19.21**	**19.21**	**18.30**	**18.35**	**16.33**	**16.33**	15.07	15.34
	Real-SN DnCNN	19.60	19.60	18.15	18.15	16.49	16.49	15.10	15.10	18.73	18.73	17.41	17.41	15.98	15.98	14.26	14.28
	Real-SN DnCNN Bspline+	17.35	17.94	14.24	14.52	15.48	15.76	12.98	13.40	16.81	17.15	12.74	12.88	14.60	15.06	11.99	12.40
	SimpleCNN	19.21	19.25	18.81	18.81	16.61	16.68	**15.89**	**15.89**	18.42	18.46	18.08	18.08	16.09	16.09	**15.45**	**15.45**
	Real-SN SimpleCNN	18.57	18.60	16.91	17.31	16.23	14.24	13.83	14.49	17.74	17.79	16.42	16.70	15.75	15.77	13.37	14.02
	Real-SN SimpleCNN Bspline	19.05	19.05	17.55	17.55	16.57	16.57	14.81	14.81	18.13	18.21	16.91	16.91	16.09	16.10	14.34	14.34
PnP-ADMM	DnCNN	**19.80**	**19.91**	**19.83**	**19.90**	16.99	17.03	**17.09**	**17.11**	**18.85**	**19.10**	**18.91**	**19.09**	**16.61**	**16.72**	**16.61**	**16.67**
	Real-SN DnCNN	19.75	19.85	19.10	19.56	**17.10**	**17.10**	16.47	16.76	18.75	18.92	17.98	18.56	16.55	16.55	15.94	16.38
	Real-SN DnCNN Bspline	19.14	19.48	19.21	19.21	16.34	16.42	16.83	16.85	18.26	18.52	18.39	18.51	16.18	16.26	16.27	16.34
	SimpleCNN	19.49	19.49	19.44	19.45	16.96	16.96	16.95	16.95	18.54	18.64	18.45	18.57	16.44	16.46	16.43	16.46
	Real-SN SimpleCNN	18.79	18.86	16.87	17.00	16.48	16.48	14.69	14.77	17.79	17.90	15.96	16.09	15.93	16.00	13.92	13.99
	Real-SN SimpleCNN Bspline	18.74	18.77	17.11	17.21	16.52	16.52	14.97	15.04	17.69	17.75	16.28	16.37	15.97	16.02	14.28	14.32

Sampling approach		Cartesian							
Image		Brain				Bust			
CNN architecture		R	R*	H	H*	R	R*	H	H*
Zero-filling		8.67				6.03			
PnP-FBS	DnCNN	**15.00**	**15.00**	14.27	14.27	14.22	14.22	13.43	13.52
	Real-SN DnCNN	14.19	14.43	13.70	13.70	13.54	13.58	12.87	12.87
	Real-SN DnCNN Bspline+	13.55	13.67	9.51	10.64	12.22	12.73	9.20	9.38
	SimpleCNN	14.82	14.85	**14.55**	**14.55**	**14.30**	**14.30**	**13.72**	**13.72**
	Real-SN SimpleCNN	14.38	14.46	12.99	13.13	13.82	13.84	11.46	11.75
	Real-SN SimpleCNN Bspline	14.80	14.80	13.15	13.15	13.80	13.94	11.92	11.92
PnP-ADMM	DnCNN	**15.07**	**15.19**	**14.70**	**15.13**	**13.99**	**14.19**	13.61	13.74
	Real-SN DnCNN	14.74	14.99	14.32	15.03	13.75	13.93	13.63	13.80
	Real-SN DnCNN Bspline	14.21	14.59	14.49	14.49	12.82	13.33	13.39	13.41
	SimpleCNN	14.65	14.97	14.67	14.92	13.82	14.06	**13.82**	**14.08**
	Real-SN SimpleCNN	13.83	13.96	12.57	12.67	12.73	13.00	10.63	10.79
	Real-SN SimpleCNN Bspline	13.80	13.83	12.71	12.79	12.85	13.14	11.0 0	11.16

PnP results for network with bias and scaling (*)