

Proximal Neural Networks: Wedding Variational Methods and Artificial Intelligence

III – Neural networks

Audrey REPETTI[†], Nelly PUSTELNIK[◊], Jean-Christophe PESQUET*

* CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvettes, France

◊ CNRS, ENS Lyon, Lyon, France

[†] Heriot-Watt University & Maxwell Institute for Mathematical Sciences, Edinburgh, UK

TUTORIAL – EUSIPCO 2025 – Palermo, Italy

Unified framework

Inference framework: feed-forward NN

$$\begin{aligned} (\forall \mathbf{x}^{[0]} \in \mathbb{R}^{N_0}) \quad \mathbf{x}^{[K]} &= \mathfrak{L}_{\Theta}^K(\mathbf{x}^{[0]}) \\ &= \mathfrak{T}_{\Theta_K} \circ \dots \circ \mathfrak{T}_{\Theta_1}(\mathbf{x}^{[0]}), \end{aligned}$$

Layer/iteration

$$\mathfrak{T}_{\Theta_k}: \mathbb{R}^{N_{k-1}} \rightarrow \mathbb{R}^{N_k}: \mathbf{x} \mapsto \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \mathbf{b}_k),$$

- ▶ $\mathbf{L}_k: \mathbb{R}^{N_{k-1}} \rightarrow \mathbb{R}^{N_k}$: linear operator,
- ▶ $\mathbf{b}_k \in \mathbb{R}^{N_k}$: shift parameter,
- ▶ $\mathfrak{D}_{\Lambda_k}: \mathbb{R}^{N_k} \rightarrow \mathbb{R}^{N_k}$: nonlinear operator parametrized by Λ_k .

Parameters: $\Theta = \cup_{k=1}^K \Theta_k$ with $\Theta_k = \{\Lambda_k, \mathbf{L}_k, \mathbf{b}_k\}$.

Training

DATASET : $\mathcal{S} = \{(\bar{\mathbf{x}}_j, \mathbf{z}_j) \in \mathbb{R}^N \times \mathbb{R}^M \mid j \in \mathbb{I} \cup \mathbb{J}\}$ pairs of (groundtruth, observation)

- *training set* $(\bar{\mathbf{x}}_j, \mathbf{z}_j)_{j \in \mathbb{I}}$ with size $|\mathbb{I}|$
- *testing set* $(\bar{\mathbf{x}}_j, \mathbf{z}_j)_{j \in \mathbb{J}}$ with size $|\mathbb{J}|$

EXAMPLE: $\mathbf{z}_j = \mathbf{A}\bar{\mathbf{x}}_j + \varepsilon_j$



Training

DATASET : $\mathcal{S} = \{(\bar{\mathbf{x}}_j, \mathbf{z}_j) \in \mathbb{R}^N \times \mathbb{R}^M \mid j \in \mathbb{I} \cup \mathbb{J}\}$ pairs of (groundtruth, observation)

- *training set* $(\bar{\mathbf{x}}_j, \mathbf{z}_j)_{j \in \mathbb{I}}$ with size $|\mathbb{I}|$
- *testing set* $(\bar{\mathbf{x}}_j, \mathbf{z}_j)_{j \in \mathbb{J}}$ with size $|\mathbb{J}|$

TRAINING STAGE: find $\hat{\Theta} \in \operatorname{Argmin}_{\Theta} \frac{1}{|\mathbb{I}|} \sum_{j \in \mathbb{I}} \ell(\bar{\mathbf{x}}_j, \mathfrak{L}_{\Theta}^K(\mathbf{z}_j)).$

- ℓ : loss function
- High-dimensional non-convex optimization problem
- Use of auto-differentiation in standard toolboxes (e.g., Pytorch).

Training

DATASET : $\mathcal{S} = \{(\bar{\mathbf{x}}_j, \mathbf{z}_j) \in \mathbb{R}^N \times \mathbb{R}^M \mid j \in \mathbb{I} \cup \mathbb{J}\}$ pairs of (groundtruth, observation)

- *training set* $(\bar{\mathbf{x}}_j, \mathbf{z}_j)_{j \in \mathbb{I}}$ with size $|\mathbb{I}|$
- *testing set* $(\bar{\mathbf{x}}_j, \mathbf{z}_j)_{j \in \mathbb{J}}$ with size $|\mathbb{J}|$

TRAINING STAGE: find $\hat{\Theta} \in \operatorname{Argmin}_{\Theta} \frac{1}{|\mathbb{I}|} \sum_{j \in \mathbb{I}} \ell(\bar{\mathbf{x}}_j, \mathfrak{L}_{\Theta}^K(\mathbf{z}_j)).$

- ℓ : loss function
- High-dimensional non-convex optimization problem
- Use of auto-differentiation in standard toolboxes (e.g., Pytorch).

EVALUATION: A properly trained network must satisfy

$$(\forall j \in \mathbb{J}) \quad \bar{\mathbf{x}}_j \approx \mathfrak{L}_{\Theta}^K(\mathbf{z}_j)$$

NN example: Multi-Layer-Perceptron (MLP)

DEFINITION: [Virmaux, Scaman, 2018]

A K -layer MLP is of the form of

$$\mathfrak{L}_{\Theta}^K = \mathfrak{T}_{\Theta_1} \circ \dots \circ \mathfrak{T}_{\Theta_K} \quad \text{with} \quad \mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \mathbf{b}_k),$$

where $(\mathbf{L}_k)_{1 \leq k \leq K}$ are dense matrices (i.e., fully connected layers).

REMARK: When $K = 1$, the network is then referred to as a perceptron.

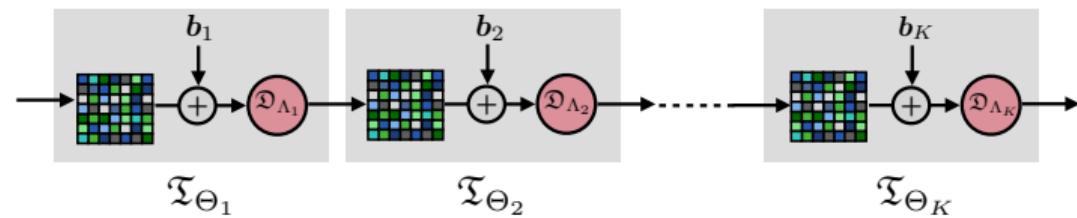


Nonlinearity



Dense matrix

MLP



NN example: DnCNN

DEFINITION: DnCNN architecture takes the form

$$\mathcal{L}_{\Theta}^K = \mathfrak{T}_{\Theta_1} \circ \dots \circ \mathfrak{T}_{\Theta_K} \quad \text{with} \quad \mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + b_k)$$

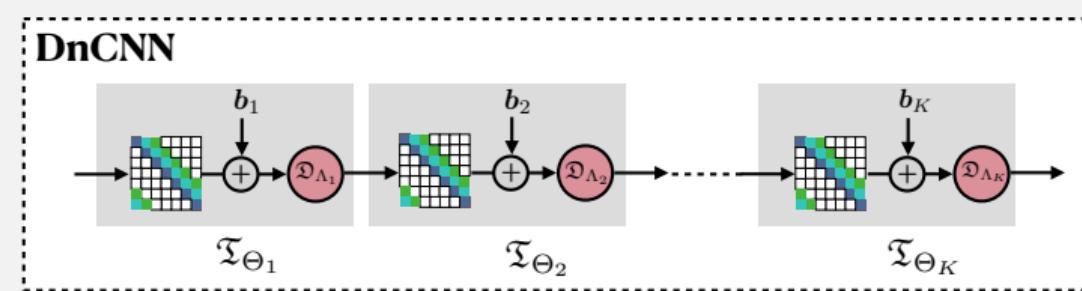
- \mathfrak{D}_{Λ_1} : ReLU activation function
- $(\mathfrak{D}_{\Lambda_k})_{2 \leq k \leq K-1}$: compositions of batch normalization and ReLU
- \mathfrak{D}_{Λ_K} : identity
- $(\mathbf{L}_k)_{1 \leq k \leq K}$: convolution operators

REMARK: Introduced as a denoising network [Zhang et al., 2017]

 Nonlinearity

 Dense matrix

 Convolution matrix



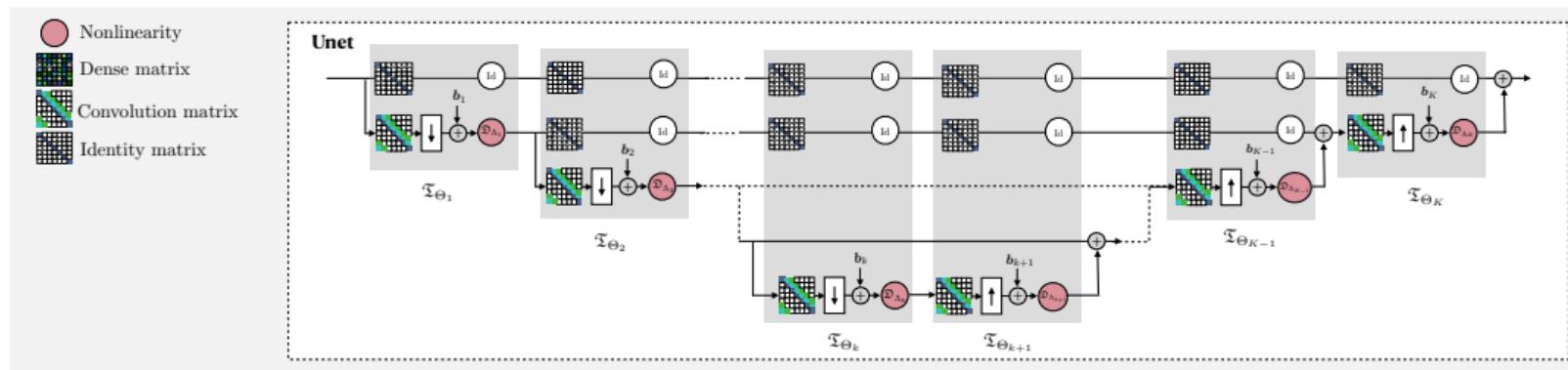
NN example: U-Net

DEFINITION: Unet is an encoder-decoder architecture defined by

$$\mathcal{L}_{\Theta}^K = \mathfrak{T}_{\Theta_1} \circ \dots \circ \mathfrak{T}_{\Theta_K} \quad \text{with} \quad \mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \mathbf{b}_k)$$

- $(\mathbf{L}_k)_{1 \leq k \leq K}$ are convolution operators
- $(\mathfrak{D}_{\Lambda_k})_{1 \leq k \leq K}$ are a combination of rectified linear units (ReLU) and max-pool/up-conv for upsampling/downsampling operations

REMARK: Developed for image segmentation [Tonneberger et al., 2015]



NN example: U-Net

DEFINITION: Unet is an encoder-decoder architecture defined by

$$\mathcal{L}_{\Theta}^K = \mathfrak{T}_{\Theta_1} \circ \dots \circ \mathfrak{T}_{\Theta_K} \quad \text{with} \quad \mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \mathbf{b}_k)$$

- $(\mathbf{L}_k)_{1 \leq k \leq K}$ are convolution operators
- $(\mathfrak{D}_{\Lambda_k})_{1 \leq k \leq K}$ are a combination of rectified linear units (ReLU) and max-pool/up-conv for upsampling/downsampling operations

REMARK: Unet architectures often use **skip-connections**:

- Considering a layer k where the variable needs to be “skipped”:

$$\mathfrak{T}_{\Theta_k}(\mathbf{x}) = \begin{pmatrix} \mathbf{x} \\ \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \mathbf{x} + \mathbf{b}_k) \end{pmatrix}$$

- Skip-connections and inertia/momentum share the same idea of reusing information from a past layer/iteration.

Outline

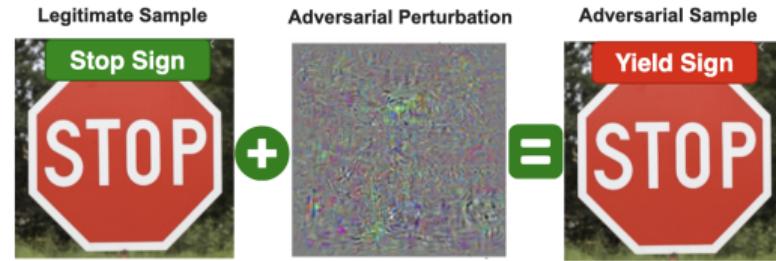
ROBUSTNESS: Lipschitz constant

LINK BETWEEN NEURAL NETWORKS AND PROXIMITY OPERATORS

- Activation functions and proximity operators
- Activation, potential and penalization operators

THEORETICAL MOTIVATION: ROBUSTNESS OF NNs

Robustness & Lipschitz constant

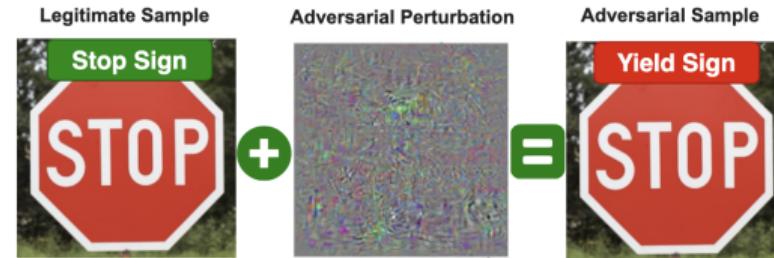


[Ahmad et al. (2022)]

QUESTION: How to control robustness of NNs?

IDEA: Use Lipschitz constant [Hoffman et al. (2019)] [Jakubovitz and Giryes (2018)] [Pesquet et al. (2021)]

Robustness & Lipschitz constant



[Ahmad et al. (2022)]

QUESTION: How to control robustness of NNs?

IDEA: Use Lipschitz constant [Hoffman et al. (2019)] [Jakubovitz and Giryes (2018)] [Pesquet et al. (2021)]

DEFINITION: Let an input $x \in \mathbb{R}^{N_0}$ and a perturbation $\epsilon \in \mathbb{R}^{N_0}$.

The error on the output can be bounded as: $\|\mathcal{L}_\Theta^K(x + \epsilon) - \mathcal{L}_\Theta^K(x)\| \leq \omega \|\epsilon\|$

REMARKS: • $\omega > 0$ is (by definition) a Lipschitz constant of \mathcal{L}_Θ^K

- ω : certificate of the network robustness

- Important and complementary to the performance in terms of estimation quality

Robustness: Control of the Lipschitz constant

DEFINITION: Let an input $\mathbf{x} \in \mathbb{R}^{N_0}$ and a perturbation $\epsilon \in \mathbb{R}^{N_0}$.

The error on the output can be bounded as: $\|\mathfrak{L}_\Theta^K(\mathbf{x} + \epsilon) - \mathfrak{L}_\Theta^K(\mathbf{x})\| \leq \omega \|\epsilon\|$

- **LAYER-BY-LAYER CERTIFICATE:**

Let $\mathfrak{L}_\Theta^K = \mathfrak{T}_{\Theta_1} \circ \dots \circ \mathfrak{T}_{\Theta_K}$, with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \mathbf{b}_k)$. If \mathfrak{D}_{Λ_k} is nonexpansive, then a

Lipschitz constant for \mathfrak{L}_Θ^K is $\omega = \prod_{k=1}^K \|\mathbf{L}_k\|_S$

- **GLOBAL CERTIFICATE:**

If \mathfrak{L}_Θ^K is differentiable, its Lipschitz constant is given by $\omega = \sup_{\mathbf{x} \in \mathbb{R}^{N_0}} \|\nabla \mathfrak{L}_\Theta^K(\mathbf{x})\|_S$ where

$$\nabla \mathfrak{L}_\Theta^K(\mathbf{x}) = \left(\frac{\partial_i \mathfrak{L}_\Theta^K(\mathbf{x})}{\partial \mathbf{x}} \right)_{1 \leq i, j \leq N_0} \text{ is the Jacobian of } \mathfrak{L}_\Theta^K \text{ at } \mathbf{x} \in \mathbb{R}^{N_0}$$

Layer-by-layer certificate

- **LAYER-BY-LAYER CERTIFICATE:**

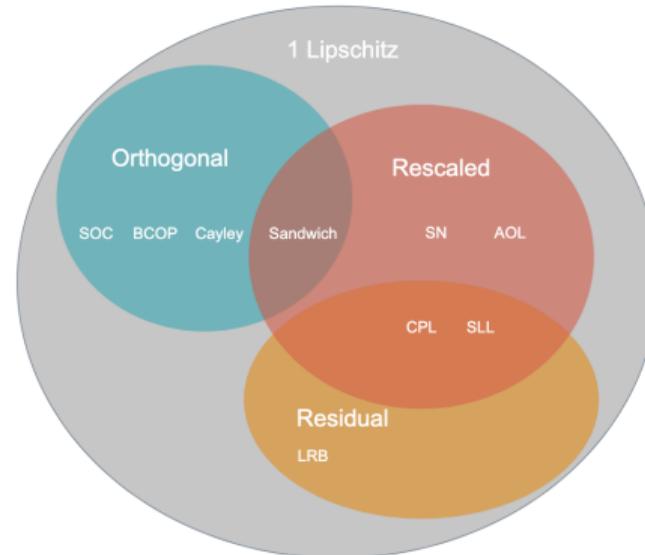
Let $\mathfrak{L}_\Theta^K = \mathfrak{T}_{\Theta_1} \circ \dots \circ \mathfrak{T}_{\Theta_K}$, with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \mathbf{b}_k)$. If \mathfrak{D}_{Λ_k} is nonexpansive, then a

Lipschitz constant for \mathfrak{L}_Θ^K is $\omega = \prod_{k=1}^K \|\mathbf{L}_k\|_S$

REMARKS:

- Bound not tight
- Bounds can often be overly pessimistic in practice, thus providing limited insights into the robustness of the network [Combettes, Pesquet (2020)][Neacsu et al. (2024)].

Some methods to control the Lipschitz constant of NN layers



[Delattre et al. (2025)]

Fig. 2.1.: Venn diagram illustrating the categorization of Lipschitz layers, including scaled layers, orthogonal layers, and residual layers.

Global certificate

- **GLOBAL CERTIFICATE:**

If \mathfrak{L}_Θ^K is differentiable, its Lipschitz constant is given by $\omega = \sup_{\mathbf{x} \in \mathbb{R}^{N_0}} \|\nabla \mathfrak{L}_\Theta^K(\mathbf{x})\|_S$

REMARKS:

- Impossible to compute over all images in $\mathbb{R}^N \rightsquigarrow$ Known to be an NP-hard problem
- Restrict the computation to a set of images $(\mathbf{x}_i)_{i \in \mathbb{I}}$ [Pesquet et al. 2021], i.e.

$$\omega \approx \max_{(\mathbf{x}_i)_{i \in \mathbb{I}}} \|\nabla \mathfrak{L}_\Theta^K(\mathbf{x}_i)\|_S$$

- \rightsquigarrow Underestimation of the Lipschitz constant.
- \rightsquigarrow Computed using power iterations coupled with automatic differentiation.
- \rightsquigarrow Used for constraining the value of ω [Pesquet et al. 2021] (*see section PnP*)
- \rightsquigarrow Used to compare robustness of different architectures [Le et al. 2023] (*see section unfolded*)

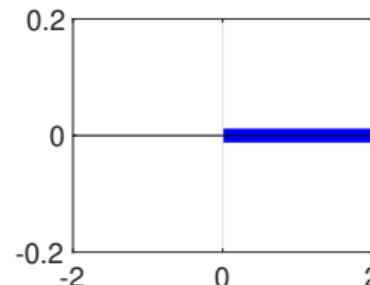
LINKS BETWEEN NNs AND PROXIMITY OPERATORS

Activation functions and proximity operators

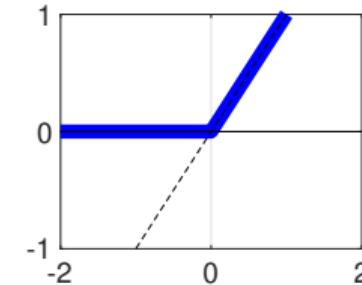
REMARK: Most activation functions used in imaging are proximity operators
[Combettes & Pesquet, 2020]

EXAMPLES: ReLu

Potential
 $\varphi(x) = \iota_{[0, +\infty)}$



Activation/ proximity operator
 $\mathfrak{D}_\Lambda(x) = \text{prox}_\varphi(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$



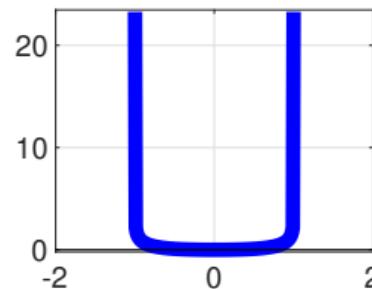
Activation functions and proximity operators

REMARK: Most activation functions used in imaging are proximity operators
[Combettes & Pesquet, 2020]

EXAMPLES: Arctangent

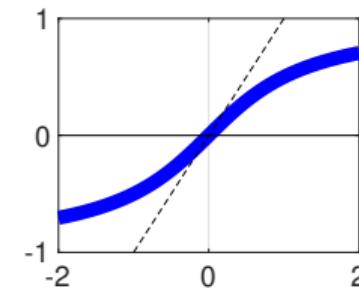
Potential

$$\varphi(x) = \begin{cases} -\frac{2}{\pi} \ln \left(\cos \left(\frac{\pi x}{2} \right) \right) - \frac{x^2}{2} & \text{if } |x| < 1 \\ +\infty & \text{otherwise} \end{cases}$$



Activation/ proximity operator

$$\mathfrak{D}_\Lambda(x) = \text{prox}_\varphi(x) = \frac{2 \arctan(x)}{\pi}$$



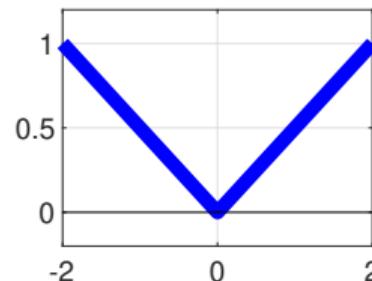
Activation functions and proximity operators

REMARK: Most activation functions used in imaging are proximity operators
[Combettes & Pesquet, 2020]

EXAMPLES: Soft thresholding

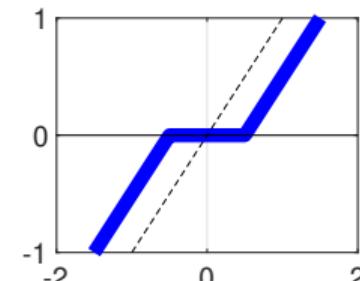
Potential

$$\varphi(x) = \lambda|x|$$



Activation/ proximity operator

$$\mathfrak{D}_\Lambda(x) = \text{prox}_\varphi(x) = \text{sign}(x) \max\{|x| - \lambda, 0\}$$



Link between activation functions and proximity operators

PROPOSITION [Combettes & Pesquet (2020)] [Gharbi et al. (2023)]

Let $\varrho: \mathcal{H} \rightarrow \mathcal{H}$ be a function deriving from a potential, i.e., there exists a **differentiable** function $\psi: \mathcal{H} \rightarrow \mathbb{R}$ such that $\varrho = \nabla \psi$.

- Let $\beta \in (0, +\infty)$ and let $\mu \in (-1, +\infty)$ be such that $\beta = 1/(1 + \mu)$.
Then ϱ is the proximity operator of a μ -convex function φ (i.e., $\varrho = \text{prox}_\varphi$)
 $\Leftrightarrow \psi$ is convex with a β - Lipschitz continuous gradient

REMINDER: • φ is μ -convex if $\varphi - \mu \frac{\|\cdot\|^2}{2}$ is convex

Link between activation functions and proximity operators

PROPOSITION [Combettes & Pesquet (2020)] [Gharbi et al. (2023)]

Let $\varrho: \mathcal{H} \rightarrow \mathcal{H}$ be a function deriving from a potential, i.e., there exists a **differentiable** function $\psi: \mathcal{H} \rightarrow \mathbb{R}$ such that $\varrho = \nabla \psi$.

- Let $\beta \in (0, +\infty)$ and let $\mu \in (-1, +\infty)$ be such that $\beta = 1/(1 + \mu)$.

Then ϱ is the proximity operator of a μ -convex function φ (i.e., $\varrho = \text{prox}_\varphi$)

$\Leftrightarrow \psi$ is convex with a β - Lipschitz continuous gradient

- ϱ is α -averaged with $\alpha \in [1/2, 1]$

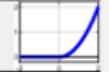
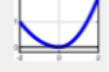
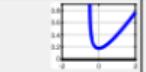
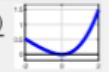
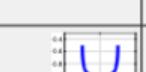
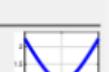
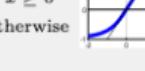
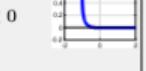
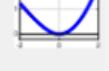
\Leftrightarrow it is a (2α) over-relaxation of the proximity operator of a convex function φ , that is

$$(\forall x \in \mathcal{H}) \quad \varrho(x) = x + 2\alpha(\text{prox}_\varphi x - x)$$

REMINDER: • φ is μ -convex if $\varphi - \mu \frac{\|\cdot\|^2}{2}$ is convex

• ϱ is α -averaged if $(1 - \alpha^{-1})\text{Id} + \alpha^{-1}\varrho$ is nonexpansive

Activation functions, potentials and penalizations: Some examples

Name	Prox/Activation $x \in \mathbb{R} \mapsto \varrho(x) = \text{prox}_\varphi(x) = \psi'(x)$	Potential $x \in \mathbb{R} \mapsto \varphi(x)$	Penalization $x \in \mathbb{R} \mapsto \psi(x)$
ReLU	$\begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$ 	$\iota_{[0, +\infty)}$ 	$\begin{cases} \frac{x^2}{2} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$ 
Parametric ReLU	$\begin{cases} x & \text{if } x > 0 \\ \lambda x & \text{otherwise} \end{cases}$ 	$\begin{cases} 0 & \text{if } x > 0 \\ \frac{(1/\lambda - 1)x^2}{2} & \text{otherwise} \end{cases}$ 	$\begin{cases} \frac{x^2}{2} & \text{if } x > 0 \\ \lambda \frac{x^2}{2} & \text{otherwise} \end{cases}$ 
Bent identity	$\frac{x + \sqrt{x^2 + 1} - 1}{2}$ 	$\begin{cases} \frac{\pi}{2} - \frac{\ln(x + 1/2)}{4} & \text{if } x > -1/2 \\ +\infty & \text{otherwise} \end{cases}$ 	$\frac{x^2 + x\sqrt{x^2 + 1} - 2x + \text{arcsinh}(x)}{4}$ 
Inverse square root	$\frac{x}{\sqrt{1+x^2}}$ 	$\begin{cases} -\frac{x^2}{2} - \sqrt{1-x^2} & \text{if } x \leq 1 \\ +\infty & \text{otherwise} \end{cases}$ 	$\sqrt{1+x^2}$ 
Inverse square root linear unit	$\begin{cases} x & \text{if } x \geq 0 \\ \frac{x}{\sqrt{1+x^2}} & \text{otherwise} \end{cases}$ 	$\begin{cases} 0 & \text{if } x \geq 0 \\ 1 - \frac{x^2}{2} - \sqrt{1-x^2} & \text{if } -1 \leq x < 0 \\ +\infty & \text{otherwise} \end{cases}$ 	$\begin{cases} \frac{x^2}{2} & \text{if } x \geq 0 \\ \sqrt{1+x^2} - 1 & \text{otherwise} \end{cases}$ 
Arctangent	$\frac{2 \arctan(x)}{\pi}$ 	$\begin{cases} -\frac{2}{\pi} \ln \left(\cos \left(\frac{\pi x}{2} \right) \right) - \frac{x^2}{2} & \text{if } x < 1 \\ +\infty & \text{otherwise} \end{cases}$ 	$\frac{2 \arctan(x)x - \ln(x^2 + 1)}{\pi}$ 
Unimodal sigmoid	$\frac{1}{1 + \exp(-x)} - \frac{1}{2}$ 	$\begin{cases} (\frac{1}{2} + x) \ln(\frac{1}{2} + x) & \text{if } x < 1/2 \\ -\frac{1}{2}(x^2 + \frac{1}{4}) & \text{if } x = 1/2 \\ +\infty & \text{otherwise} \end{cases}$ 	$\frac{\pi}{2} + \ln(1 + \exp(-x))$ 

Activation functions & proximity operators: Weakly convex case

PROPOSITION [Hurault et al. (2022)]

Let $\beta \in [0, 1]$ and let $\psi: \mathcal{H} \rightarrow \mathbb{R}$ be a differentiable function with a β -Lipschitz gradient. Then, there exists a $\beta/(\beta + 1)$ -weakly convex function $\varphi: \mathcal{H} \rightarrow (-\infty, +\infty]$, such that

$$\text{Id} - \nabla\psi = \text{prox}_\varphi$$

REMARK: If φ is convex, proper and l.s.c., then ψ is the Moreau envelope of φ defined as

$$(\forall \tilde{\mathbf{x}} \in \mathcal{H}) \quad \psi(\tilde{\mathbf{x}}) = \inf_{\mathbf{x} \in \mathcal{H}} \varphi(\mathbf{x}) + \frac{1}{2} \|\mathbf{x} - \tilde{\mathbf{x}}\|^2.$$

Conclusion

