# Proximal Neural Networks: Marrying 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**

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

**Layer/iteration**

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

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

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

# Training

DATASET : $\mathcal{S} = \left\{(\overline{\boldsymbol{x}}_j, \boldsymbol{z}_j) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; j \in \mathbb{I} \cup \mathbb{J}\right\}$ pairs of (groundtruth, observation)

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

EXEMPLE: $\boldsymbol{z}_j = \mathrm{A}\overline{\boldsymbol{x}}_j + \varepsilon_j$

$\overline{\mathrm{x}}_\ell$

$\mathrm{z}_\ell$

# Training

DATASET : $\mathcal{S} = \left\{(\overline{\boldsymbol{x}}_j, \boldsymbol{z}_j) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; j \in \mathbb{I} \cup \mathbb{J}\right\}$ pairs of (groundtruth, observation)

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

TRAINING STAGE: find $\widehat{\Theta} \in \underset{\Theta}{\operatorname{Argmin}} \; \frac{1}{|\mathbb{I}|} \sum_{j\in\mathbb{I}} \ell\big(\overline{\boldsymbol{x}}_j, \mathfrak{L}_\Theta^K(\boldsymbol{z}_j)\big)$.

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

# Training

DATASET : $\mathcal{S} = \left\{(\overline{\boldsymbol{x}}_j, \boldsymbol{z}_j) \in \mathbb{R}^N \times \mathbb{R}^M \;\middle|\; j \in \mathbb{I} \cup \mathbb{J}\right\}$ pairs of (groundtruth, observation)

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

TRAINING STAGE: find $\widehat{\Theta} \in \underset{\Theta}{\operatorname{Argmin}} \; \frac{1}{|\mathbb{I}|} \sum_{j\in\mathbb{I}} \ell\big(\overline{\boldsymbol{x}}_j, \mathfrak{L}_\Theta^K(\boldsymbol{z}_j)\big)$.

- $\ell$: 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

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

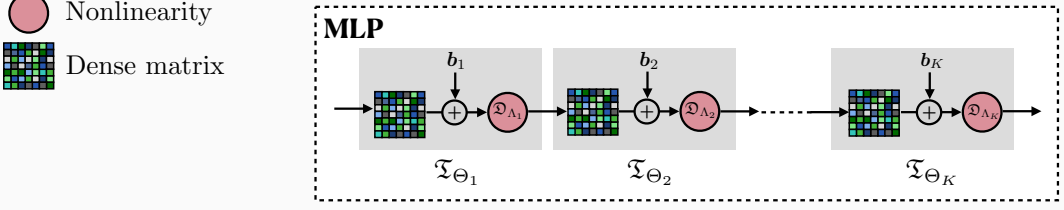
# NN example: Multi-Layer-Perceptron (MLP)

**Definition:** [Scaman, Virmaux, 2018]
A $K$-layer MLP is of the form of

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

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.



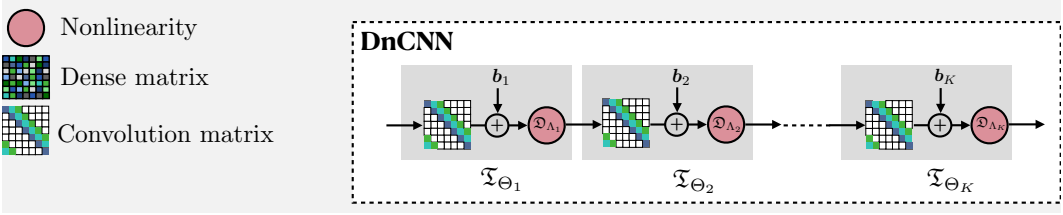
# NN example: DnCNN

**Definition:** DnCNN architecture takes the form

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

- $\mathfrak{D}_{\Lambda_1}$: ReLU activation function
- $(\mathfrak{D}_{\Lambda_k})_{2 \leq k \leq K-1}$: compositions of batch normalization and ReLU
- $\mathfrak{D}_{\Lambda_K}$: identity
- $(\mathbf{L}_k)_{1 \leq k \leq K}$: convolution operators

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



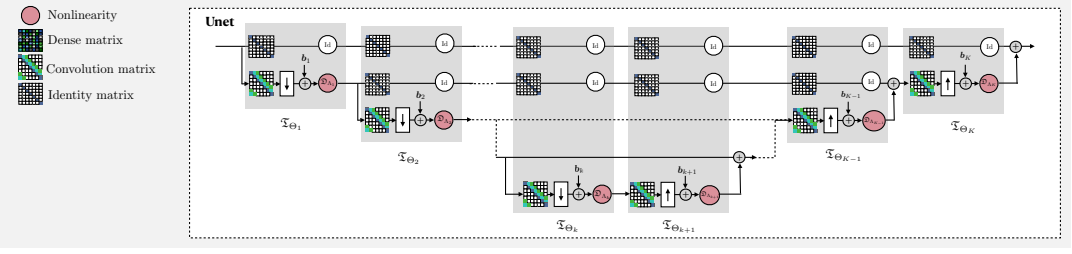
# NN example: U-Net

**Definition:** Unet is an encoder-decoder architecture defined by

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

- $(\mathbf{L}_k)_{1 \leq k \leq K}$ 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 [Ronneberger et al., 2015]



# NN example: U-Net

**Definition:** Unet is an encoder-decoder architecture defined by

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

- $(\mathbf{L}_k)_{1 \leq k \leq K}$ 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":

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

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

# Outline

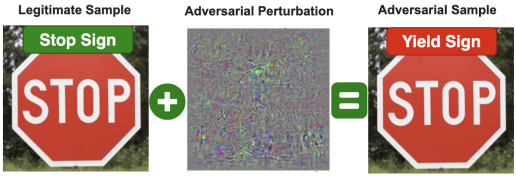
Robustness: Lipschitz constant

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

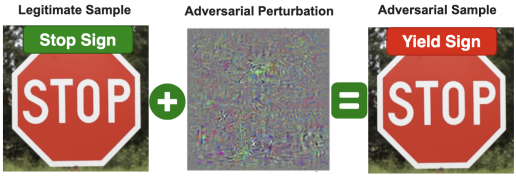



[ Ahmad et al. (2022)]

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 $\mathrm{x} \in \mathbb{R}^{N_0}$ and a perturbation $\epsilon \in \mathbb{R}^{N_0}$.
The error on the output can be bounded as: $\|\mathfrak{L}_{\Theta}^{K}(\boldsymbol{x}+\epsilon) - \mathfrak{L}_{\Theta}^{K}(\boldsymbol{x})\| \leq \omega\|\epsilon\|$

REMARKS:
- $\omega > 0$ is (by definition) a Lipschitz constant of $\mathfrak{L}_{\Theta}^{K}$
- $\omega$: 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 $\mathrm{x} \in \mathbb{R}^{N_0}$ and a perturbation $\epsilon \in \mathbb{R}^{N_0}$.
The error on the output can be bounded as: $\|\mathfrak{L}_\Theta^K(\boldsymbol{x}+\epsilon) - \mathfrak{L}_\Theta^K(\boldsymbol{x})\| \leq \omega\|\epsilon\|$

- LAYER-BY-LAYER CERTIFICATE:

  Let $\mathfrak{L}_\Theta^K = \mathfrak{T}_{\Theta_1} \circ \ldots \circ \mathfrak{T}_{\Theta_K}$, with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \boldsymbol{b}_k)$. If $\mathfrak{D}_{\Lambda_k}$ is nonexpansive, then a Lipschitz constant for $\mathfrak{L}_\Theta^K$ is $\omega = \prod_{k=1}^{K} \|\mathbf{L}_k\|_{\mathrm{S}}$

- GLOBAL CERTIFICATE:

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

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

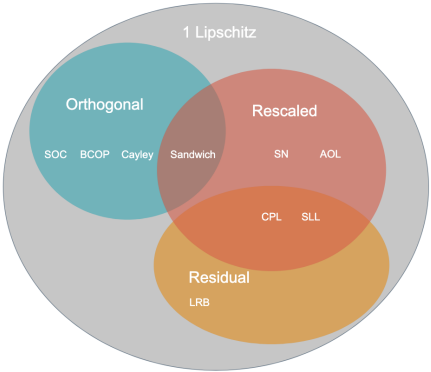
# Layer-by-layer certificate

- **Layer-by-layer certificate:**
  Let $\mathfrak{L}_\Theta^K = \mathfrak{T}_{\Theta_1} \circ \ldots \circ \mathfrak{T}_{\Theta_K}$, with $\mathfrak{T}_{\Theta_k} = \mathfrak{D}_{\Lambda_k}(\mathbf{L}_k \cdot + \boldsymbol{b}_k)$. If $\mathfrak{D}_{\Lambda_k}$ is nonexpansive, then a Lipschitz constant for $\mathfrak{L}_\Theta^K$ is $\omega = \prod_{k=1}^{K} \|\mathbf{L}_k\|_{\mathrm{S}}$

**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 (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 $\mathfrak{L}_\Theta^K$ is differentiable, its Lipschitz constant is given by $\omega = \sup_{\boldsymbol{x} \in \mathbb{R}^{N_0}} \|\boldsymbol{\nabla} \mathfrak{L}_\Theta^K(\boldsymbol{x})\|_{\mathrm{S}}$

**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 $(\mathrm{x}_i)_{i \in \mathbb{I}}$ [Pesquet et al. 2021], i.e.

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

  - $\rightsquigarrow$ Underestimation of the Lipschitz constant.
  - $\rightsquigarrow$ Computed using power iterations coupled with automatic differentiation.
  - $\rightsquigarrow$ Used for constraining the value of $\omega$ [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

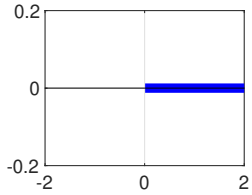
# Activation functions and proximity operators

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

EXAMPLES: ReLu

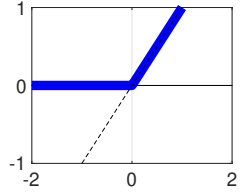
Potential

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




Activation/ proximity operator

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



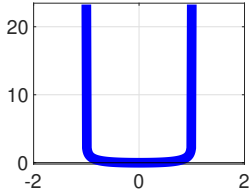
# Activation functions and proximity operators

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

EXAMPLES: Arctangent

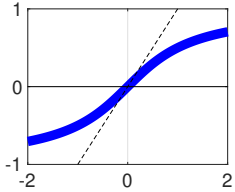
Potential

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




Activation/ proximity operator

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

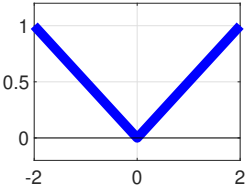


# Activation functions and proximity operators

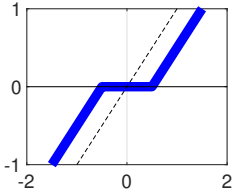
REMARK: Most activation functions used in imaging are proximity operators
[Combettes & Pesquet, 2020]

EXAMPLES: Soft thresholding

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




Activation/ proximity operator
$\mathfrak{D}_\Lambda(x) = \mathrm{prox}_\varphi(x) = \mathrm{sign}(x)\max\{|x| - \lambda, 0\}$



# Link between activation functions and proximity operators

PROPOSITION [Combettes & Pesquet, 2020] [Gharbi *et al.*, 2023]
Let $\varrho\colon \mathcal{H} \to \mathcal{H}$ be a function deriving from a potential, i.e., there exists a differentiable function $\psi\colon \mathcal{H} \to \mathbb{R}$ such that $\varrho = \nabla\psi$.

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

REMINDER: • $\varphi$ is $\mu$-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 $\varrho\colon \mathcal{H} \to \mathcal{H}$ be a function deriving from a potential, i.e., there exists a differentiable function $\psi\colon \mathcal{H} \to \mathbb{R}$ such that $\varrho = \nabla\psi$.

- Let $\beta \in (0, +\infty)$ and let $\mu \in (-1, +\infty)$ be such that $\beta = 1/(1+\mu)$.
  Then $\varrho$ is the proximity operator of a $\mu$-convex function $\varphi$ (i.e., $\varrho = \operatorname{prox}_\varphi$)
  $\Leftrightarrow$ $\psi$ is convex with a $\beta$- Lipschitz continuous gradient
- $\varrho$ is $\alpha$-averaged with $\alpha \in [1/2, 1]$
  $\Leftrightarrow$ it is a $(2\alpha)$ over-relaxation of the proximity operator of a convex function $\varphi$, that is
  $$(\forall \boldsymbol{x} \in \mathcal{H}) \quad \varrho(\boldsymbol{x}) = \boldsymbol{x} + 2\alpha\big(\operatorname{prox}_\varphi \boldsymbol{x} - \boldsymbol{x}\big)$$

REMINDER:
- $\varphi$ is $\mu$-convex if $\varphi - \mu\frac{\|\cdot\|^2}{2}$ is convex
- $\varrho$ is $\alpha$-averaged if $(1-\alpha^{-1})\mathrm{Id} + \alpha^{-1}\varrho$ is nonexpansive

# Activation functions, potentials and penalizations: Some examples

| Name | Prox/Activation $x \in \mathbb{R} \mapsto \varrho(x) = \mathrm{prox}_{\varphi}(x) = \psi'(x)$ | 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 | $\frac{x + \sqrt{x^2+1} - 1}{2}$ | $\begin{cases} \frac{x}{2} - \frac{\ln(x+1/2)}{4} & \text{if } x > -1/2 \\ +\infty & \text{otherwise} \end{cases}$ | $\frac{x^2 + x\sqrt{x^2+1} - 2x + \mathrm{arcsinh}(x)}{4}$ |
| Inverse square root | $\frac{x}{\sqrt{1+x^2}}$ | $\begin{cases} -\frac{x^2}{2} - \sqrt{1-x^2} & \text{if } \lvert x\rvert \leq 1 \\ +\infty & \text{otherwise} \end{cases}$ | $\sqrt{1+x^2}$ |
| 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 | $\frac{2\arctan(x)}{\pi}$ | $\begin{cases} -\frac{2}{\pi}\ln\left(\cos\left(\frac{\pi x}{2}\right)\right) - \frac{x^2}{2} & \text{if } \lvert x\rvert < 1 \\ +\infty & \text{otherwise} \end{cases}$ | $\frac{2\arctan(x)x - \ln(\lvert x^2+1\rvert)}{\pi}$ |
| Unimodal sigmoid | $\frac{1}{1+\exp(-x)} - \frac{1}{2}$ | $\begin{cases} (\frac{1}{2}+x)\ln(\frac{1}{2}+x) + (\frac{1}{2}-x)\ln(\frac{1}{2}-x) & \\ -\frac{1}{2}(x^2+\frac{1}{4}) & \text{if } \lvert x\rvert < 1/2 \\ -1/4 & \text{if } \lvert x\rvert = 1/2 \\ +\infty & \text{otherwise} \end{cases}$ | $\frac{x}{2} + \ln(1+\exp(-x))$ |

# Activation functions & proximity operators: Weakly convex case

PROPOSITION [Hurault *et al.*, 2022]
Let $\beta \in [0,1]$ and let $\psi \colon \mathcal{H} \to \mathbb{R}$ be a differentiable function with a $\beta$-Lipschitz gradient.
Then, there exists a $\beta/(\beta+1)$-weakly convex function $\varphi \colon \mathcal{H} \to (-\infty, +\infty]$, such that

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

REMARK: If $\varphi$ is convex, proper and l.s.c., then $\psi$ is the Moreau envelope of $\varphi$ defined as

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

# Conclusion

$$\underbrace{\mathfrak{D}_{\Lambda_K}(\mathbf{L}_K \cdot + \boldsymbol{b}_K)} \quad \underbrace{\mathfrak{D}_{\Lambda_1}(\mathbf{L}_1 \cdot + \boldsymbol{b}_1)}$$

$$\underbrace{\mathfrak{T}_{\Theta_K} \circ \ldots \circ \mathfrak{T}_{\Theta_1}}$$

$$\boldsymbol{x}^{[0]} \longrightarrow \boxed{\mathfrak{L}_\Theta^K} \longrightarrow \boldsymbol{x}^{[K]}$$




| **Variational approaches and Proximal algorithms** | |
|---|---|
| $K$ large | $\mathfrak{D}_{\Lambda_k}$ : Proximity operator |
| | |
| | |
| | |
| | |
| **End-to-end** | |
| $K$ medium (i.e. $\sim 50$) | $\mathfrak{D}_{\Lambda_k}$ : Activation function |