Approximation theory for 1-Lipschitz ResNets

Davide Murari

Department of Applied Mathematics and Theoretical Physics, University of Cambridge.

In collaboration with Takashi Furuya and Carola-Bibiane Schönlieb

davidemurari.com/onelip

dm2011@cam.ac.uk

Outline

1. 1-Lipschitz Neural Networks
2. 1-Lipschitz Networks for Robust Classification
3. Universal approximation theorem for 1-Lipschitz ResNets: Unbounded width and depth
4. Universal approximation theorem for 1-Lipschitz ResNets: Unbounded depth and fixed width

1-Lipschitz Neural Networks

Why 1-Lipschitz neural networks? $\|F(\mathbf{y}) - F(\mathbf{x})\|_2 \leq \|\mathbf{y} - \mathbf{x}\|_2$

Adversarial robustness

Constraining the Lipschitz constant leads to a reduced sensitivity to input perturbations.

Why 1-Lipschitz neural networks? $\|F(\mathbf{y}) - F(\mathbf{x})\|_2 \leq \|\mathbf{y} - \mathbf{x}\|_2$

Adversarial robustness

Constraining the Lipschitz constant leads to a reduced sensitivity to input perturbations.

Wasserstein Generative Adversarial Networks (Kantorovich-Rubinstein duality)

$$W_1(\mu, \nu) = \sup_{\substack{f:\mathcal{X}\to\mathbb{R} \\ f\ 1-\text{Lipschitz}}} \mathbb{E}_{X\sim\mu}[f(X)] - \mathbb{E}_{Y\sim\nu}[f(Y)].$$

Why 1-Lipschitz neural networks? $\|F(\mathbf{y}) - F(\mathbf{x})\|_2 \leq \|\mathbf{y} - \mathbf{x}\|_2$

Adversarial robustness

Constraining the Lipschitz constant leads to a reduced sensitivity to input perturbations.

Wasserstein Generative Adversarial Networks (Kantorovich-Rubinstein duality)

$$W_1(\mu, \nu) = \sup_{\substack{f:\mathcal{X}\to\mathbb{R} \\ f\ 1-\text{Lipschitz}}} \mathbb{E}_{X\sim\mu}[f(X)] - \mathbb{E}_{Y\sim\nu}[f(Y)].$$

Convergent fixed point iterations

If $\|F(\mathbf{y}) - F(\mathbf{x})\|_2 < \|\mathbf{y} - \mathbf{x}\|_2$ for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, then $\mathbf{x}_{k+1} = F(\mathbf{x}_k)$ admits a unique and attractive fixed point. If $T_\alpha(\mathbf{x}) = (1-\alpha)\mathbf{x} + \alpha F(\mathbf{x})$, $\alpha \in (0,1)$ and F 1-Lipschitz, then whenever $\mathbf{x}_{k+1} = T_\alpha(\mathbf{x}_k)$ has a fixed point, the sequence converges.

1-Lipschitz MLPs

- Given two Lipschitz-continuous functions $F : \mathbb{R}^h \to \mathbb{R}^c$, $G : \mathbb{R}^d \to \mathbb{R}^h$, with Lipschitz constants $\mathrm{Lip}(F)$ and $\mathrm{Lip}(G)$, respectively, the composition $H = F \circ G : \mathbb{R}^d \to \mathbb{R}^c$ is Lipschitz continuous as well, with $\mathrm{Lip}(H) \leq \mathrm{Lip}(F)\mathrm{Lip}(G)$:

$$\begin{aligned} \|H(\mathbf{y}) - H(\mathbf{x})\|_2 &= \|F(G(\mathbf{y})) - F(G(\mathbf{x}))\|_2 \leq \mathrm{Lip}(F)\|G(\mathbf{y}) - G(\mathbf{x})\|_2 \\ &\leq \mathrm{Lip}(F)\mathrm{Lip}(G)\|\mathbf{y} - \mathbf{x}\|_2, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \end{aligned}$$

1-Lipschitz MLPs

- Given two Lipschitz-continuous functions $F : \mathbb{R}^h \to \mathbb{R}^c$, $G : \mathbb{R}^d \to \mathbb{R}^h$, with Lipschitz constants $\mathrm{Lip}(F)$ and $\mathrm{Lip}(G)$, respectively, the composition $H = F \circ G : \mathbb{R}^d \to \mathbb{R}^c$ is Lipschitz continuous as well, with $\mathrm{Lip}(H) \leq \mathrm{Lip}(F)\mathrm{Lip}(G)$:

$$\begin{aligned}\|H(\mathbf{y}) - H(\mathbf{x})\|_2 = \|F(G(\mathbf{y})) - F(G(\mathbf{x}))\|_2 &\leq \mathrm{Lip}(F)\|G(\mathbf{y}) - G(\mathbf{x})\|_2 \\ &\leq \mathrm{Lip}(F)\mathrm{Lip}(G)\|\mathbf{y} - \mathbf{x}\|_2, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.\end{aligned}$$

- We can get a 1-Lipschitz feedforward network (MLP) composing 1-Lipschitz layers:

$$\mathcal{N}_\theta = A_L \circ \sigma \circ A_{L-1} \circ ... \circ \sigma \circ A_1 : \mathbb{R}^d \to \mathbb{R}^c,$$

where we need $|\sigma(s) - \sigma(t)| \leq |s - t|$, and $\|A_i\|_2 \leq 1$ for $j = 1, ..., L$. Most activation functions, such as $\tanh, \mathrm{ReLU}, \mathrm{LeakyReLU}$, sigmoid, sin are 1-Lipschitz.

1-Lipschitz ResNets are more challenging to obtain

For ResNets, it is more challenging, since the basic layers are of the form

$$\mathbb{R}^d \ni \mathbf{x} \mapsto \mathbf{x} + \tau \mathcal{F}_{\theta_i}(\mathbf{x}) = \varphi^{\tau}_{\theta_i}(\mathbf{x}) \in \mathbb{R}^d, \ \tau > 0,$$

and, for a generic $\mathcal{F}_{\theta_i} : \mathbb{R}^d \to \mathbb{R}^d$, it is hard to get better bounds than

$$\|\varphi^{\tau}_{\theta_i}(\mathbf{y}) - \varphi^{\tau}_{\theta_i}(\mathbf{x})\|_2 \leq (1 + \tau \mathrm{Lip}(\mathcal{F}_{\theta_i})) \, \|\mathbf{y} - \mathbf{x}\|_2, \ \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

We hence need to modify them slightly, or properly choose the residual map $\mathcal{F}_{\theta_i}$.

Negative gradient flows

Let $V : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable convex function. We consider vector fields of the form

$$\mathcal{F}(\mathbf{x}) = -\nabla V(\mathbf{x}).$$

Given two solution curves, $\dot{\mathbf{x}}(t) = \mathcal{F}(\mathbf{x}(t))$ and $\dot{\mathbf{y}}(t) = \mathcal{F}(\mathbf{y}(t))$, we see that

$$\frac{d}{dt}\|\mathbf{x}(t) - \mathbf{y}(t)\|_2^2 = -\left(\nabla V(\mathbf{x}(t)) - \nabla V(\mathbf{y}(t))\right)^\top (\mathbf{x}(t) - \mathbf{y}(t)) \leq 0.$$

Thus, the flow map $\phi_{\mathcal{F}}^t : \mathbb{R}^d \to \mathbb{R}^d$ defined by $\phi_{\mathcal{F}}^t(\mathbf{x}(0)) = \mathbf{x}(t)$ is 1-Lipschitz.

Non-expansive gradient flows

Gradient flows on $\mathbb{R}^d$

Consider the scalar function[a] $V_\theta(x) = 1^\top \mathrm{ReLU}^2(W\mathbf{x} + \mathbf{b})/2$. Define

$$\mathcal{F}_\theta(\mathbf{x}) = -\nabla V_\theta(\mathbf{x}) = -W^\top \mathrm{ReLU}(W\mathbf{x} + \mathbf{b}).$$

If $\dot{\mathbf{x}} = \mathcal{F}_\theta(\mathbf{x})$ and $\dot{\mathbf{y}} = \mathcal{F}_\theta(\mathbf{y})$, we have $\|\mathbf{y}(t) - \mathbf{x}(t)\|_2 \leq \|\mathbf{y}(0) - \mathbf{x}(0)\|_2$ for every $t \geq 0$.

[a] $W \in \mathbb{R}^{h\times d}$, $\mathbf{b} \in \mathbb{R}^h$, $h \in \mathbb{N}$, $\theta = (W, \mathbf{b})$, and $1 \in \mathbb{R}^h$ a vector of ones.

Non-expansive gradient flows

Gradient flows on $\mathbb{R}^d$

Consider the scalar function[a] $V_\theta(x) = 1^\top \mathrm{ReLU}^2(W\mathbf{x} + \mathbf{b})/2$. Define

$$\mathcal{F}_\theta(\mathbf{x}) = -\nabla V_\theta(\mathbf{x}) = -W^\top \mathrm{ReLU}(W\mathbf{x} + \mathbf{b}).$$

If $\dot{\mathbf{x}} = \mathcal{F}_\theta(\mathbf{x})$ and $\dot{\mathbf{y}} = \mathcal{F}_\theta(\mathbf{y})$, we have $\|\mathbf{y}(t) - \mathbf{x}(t)\|_2 \leq \|\mathbf{y}(0) - \mathbf{x}(0)\|_2$ for every $t \geq 0$.

[a] $W \in \mathbb{R}^{h\times d}$, $\mathbf{b} \in \mathbb{R}^h$, $h \in \mathbb{N}$, $\theta = (W, \mathbf{b})$, and $1 \in \mathbb{R}^h$ a vector of ones.

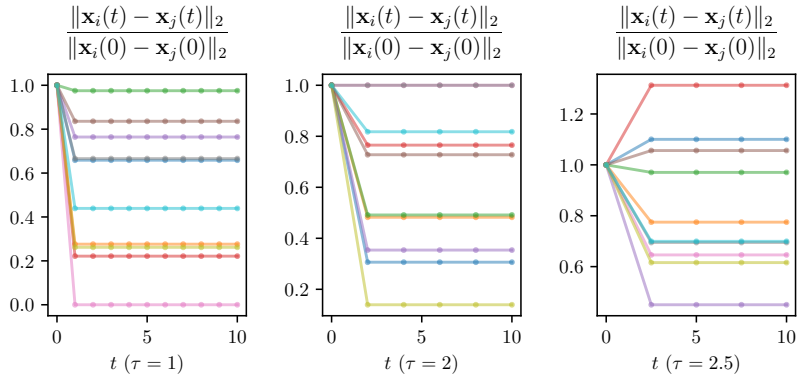
Euler step (1-Lipschitz)

If $\tau \in [0, 2/\|W\|_2^2]$, the explicit Euler map $\varphi_\theta^\tau(\mathbf{x}) = \mathbf{x} + \tau\mathcal{F}_\theta(\mathbf{x})$ is 1-Lipschitz, i.e.,

$$\|\varphi_\theta^\tau(\mathbf{y}) - \varphi_\theta^\tau(\mathbf{x})\|_2 \leq \|\mathbf{y} - \mathbf{x}\|_2, \ \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

ODEs with 1-Lipschitz solution and Euler maps

$$\dot{x}(t) = -W^\top \mathrm{ReLU}(Wx(t)), \;\; W = \frac{1}{2}\begin{bmatrix} \sqrt{2} & -\sqrt{2} \\ \sqrt{2} & \sqrt{2} \end{bmatrix}.$$



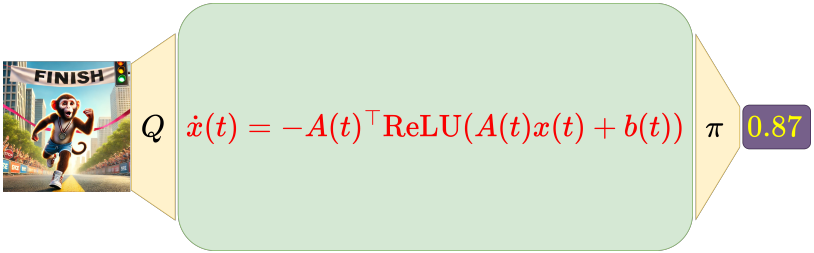
Neural networks based on gradient flows

We consider neural networks of the form

$$\mathcal{N}_\theta = \pi \circ \varphi_{\theta_L} \circ \ldots \circ \varphi_{\theta_1} \circ Q : \mathbb{R}^d \to \mathbb{R}^c,\ \varphi_{\theta_\ell} \in \mathcal{E}_h,$$

$$\mathcal{E}_h := \Big\{ \varphi : \mathbb{R}^h \to \mathbb{R}^h \ \Big|\ \varphi(\mathbf{x}) = x - \tau W^\top \mathrm{ReLU}(W\mathbf{x} + \mathbf{b}),\ W \in \mathbb{R}^{h' \times h}, \mathbf{b} \in \mathbb{R}^{h'},$$
$$h' \in \mathbb{N}, \tau \in [0, 2/\|W\|_2^2] \Big\},$$

where $Q : \mathbb{R}^d \to \mathbb{R}^h$ and $\pi : \mathbb{R}^h \to \mathbb{R}^c$ are affine maps.



1-Lipschitz Networks for Robust Classification

The problem of robust classification

Classification problem

Let $\mathcal{X} \subset \mathbb{R}^d$ be a set whose points are known to belong to C classes. Given part of their labels, we want to label the remaining points using $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^C$ where we set

$$\text{predicted class of } \mathbf{x} = \underset{c=1,\dots,C}{\arg\max} \left(\mathcal{N}_\theta(\mathbf{x})^\top \boldsymbol{e}_c \right).$$

The problem of robust classification

Classification problem

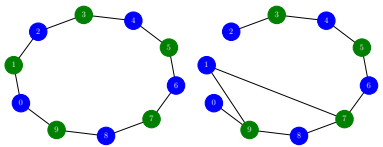
Let $\mathcal{X} \subset \mathbb{R}^d$ be a set whose points are known to belong to C classes. Given part of their labels, we want to label the remaining points using $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^C$ where we set

$$\text{predicted class of } \mathbf{x} = \underset{c=1,\dots,C}{\arg\max} \left(\mathcal{N}_\theta(\mathbf{x})^\top \boldsymbol{e}_c \right).$$

Adversarial examples

X
Label : Horse

$X + \delta$, $\|\delta\|_2 = 0.3$
Label : Dog



How to have guaranteed robustness

- Not all correct predictions are equivalent.
- Let $\ell(\mathbf{x}) = 2$ be the correct label for the point $\mathbf{x} \in \mathcal{X}$.
- $\mathcal{N}_{\theta_1}(\mathbf{x}) = \begin{bmatrix} 0.49 & 0.51 & 0 \end{bmatrix}$ is not so certain as a prediction.
- $\mathcal{N}_{\theta_2}(\mathbf{x}) = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix}$ there is a higher gap here.

How to have guaranteed robustness

- Not all correct predictions are equivalent.
- Let $\ell(\mathbf{x}) = 2$ be the correct label for the point $\mathbf{x} \in \mathcal{X}$.
- $\mathcal{N}_{\theta_1}(\mathbf{x}) = \begin{bmatrix} 0.49 & 0.51 & 0 \end{bmatrix}$ is not so certain as a prediction.
- $\mathcal{N}_{\theta_2}(\mathbf{x}) = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix}$ there is a higher gap here.

Margin: $\mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x}) := \mathcal{N}_\theta(\mathbf{x})^\top \boldsymbol{e}_{\ell(\mathbf{x})} - \max_{j \neq \ell(\mathbf{x})} \mathcal{N}_\theta(\mathbf{x})^\top \boldsymbol{e}_j.$

$$\mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x}) > 0 \implies \mathcal{N}_\theta \text{ correctly classifies } \mathbf{x}.$$

$$\mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x}) > \sqrt{2}\mathrm{Lip}(\mathcal{N}_\theta)\varepsilon \implies \mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x} + \boldsymbol{\eta}) > 0 \, \forall \|\boldsymbol{\eta}\|_2 \leq \varepsilon.$$

How to have guaranteed robustness

- Not all correct predictions are equivalent.
- Let $\ell(\mathbf{x}) = 2$ be the correct label for the point $\mathbf{x} \in \mathcal{X}$.
- $\mathcal{N}_{\theta_1}(\mathbf{x}) = \begin{bmatrix} 0.49 & 0.51 & 0 \end{bmatrix}$ is not so certain as a prediction.
- $\mathcal{N}_{\theta_2}(\mathbf{x}) = \begin{bmatrix} 0.05 & 0.9 & 0.05 \end{bmatrix}$ there is a higher gap here.

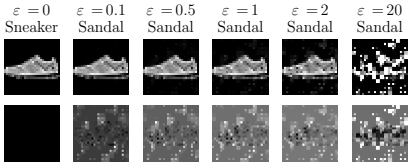
Margin: $\mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x}) := \mathcal{N}_\theta(\mathbf{x})^\top \boldsymbol{e}_{\ell(\mathbf{x})} - \max_{j \neq \ell(\mathbf{x})} \mathcal{N}_\theta(\mathbf{x})^\top \boldsymbol{e}_j.$

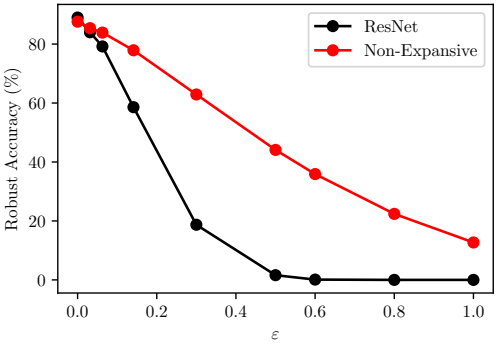
$$\mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x}) > 0 \implies \mathcal{N}_\theta \text{ correctly classifies } \mathbf{x}.$$

$$\mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x}) > \sqrt{2}\mathrm{Lip}(\mathcal{N}_\theta)\varepsilon \implies \mathcal{M}_{\mathcal{N}_\theta}(\mathbf{x} + \boldsymbol{\eta}) > 0 \, \forall \|\boldsymbol{\eta}\|_2 \leq \varepsilon.$$

- We constrain the Lipschitz constant of $\mathcal{N}_\theta$ (and train the network so it maximises the margin).

Robustness to adversarial attacks





Ferdia Sherry et al. "Designing stable neural networks using convex analysis and ODEs". In: *Physica D: Nonlinear Phenomena* 463 (2024), p. 134159

Universal approximation theorem for 1-Lipschitz ResNets: Unbounded width and depth

Statement of the theorem

Let $d \in \mathbb{N}$, $\mathcal{X} \subseteq \mathbb{R}^d$, and fix $c = 1$, i.e., consider scalar-valued networks. The networks we just derived define the following set

$$\mathcal{G}_d(\mathcal{X}) = \Big\{ \pi \circ \varphi_{\theta_L} \circ ... \circ \varphi_{\theta_1} \circ Q : \mathcal{X} \to \mathbb{R} \;\Big|\; \varphi_{\theta_\ell} \in \mathcal{E}_h,\ \ell = 1, ..., L,\ h, L \in \mathbb{N},$$
$$Q : \mathbb{R}^d \to \mathbb{R}^h,\ \pi : \mathbb{R}^h \to \mathbb{R},\ Q \text{ and } \pi \text{ affine} \Big\}.$$

We denote with $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$ the set of 1-Lipschitz functions from $\mathcal{X}$ to $\mathbb{R}$.

Statement of the theorem

Let $d \in \mathbb{N}$, $\mathcal{X} \subseteq \mathbb{R}^d$, and fix $c = 1$, i.e., consider scalar-valued networks. The networks we just derived define the following set

$$\mathcal{G}_d(\mathcal{X}) = \Big\{ \pi \circ \varphi_{\theta_L} \circ \ldots \circ \varphi_{\theta_1} \circ Q : \mathcal{X} \to \mathbb{R} \;\Big|\; \varphi_{\theta_\ell} \in \mathcal{E}_h,\ \ell = 1, \ldots, L,\ h, L \in \mathbb{N},\ Q : \mathbb{R}^d \to \mathbb{R}^h,\ \pi : \mathbb{R}^h \to \mathbb{R},\ Q \text{ and } \pi \text{ affine} \Big\}.$$

We denote with $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$ the set of 1-Lipschitz functions from $\mathcal{X}$ to $\mathbb{R}$.

Universal approximation theorem

Let $\varepsilon > 0$, $\mathcal{X} \subset \mathbb{R}^d$ compact, and $g \in \mathcal{C}_1(\mathcal{X}, \mathbb{R})$ a 1-Lipschitz function. Then, there exists $f \in \mathcal{G}_d(\mathcal{X}) \cap \mathcal{C}_1(\mathcal{X}, \mathbb{R})$ such that

$$\max_{x \in \mathcal{X}} |f(x) - g(x)| < \varepsilon.$$

Two proof techniques

We prove this theorem in two different ways:

1. First, we prove that the **Restricted Stone-Weierstrass Theorem** (see below) holds,
2. Second, we prove that each **piecewise affine 1-Lipschitz** function belongs to our set of networks, and conclude thanks to their density in the set of 1-Lipschitz functions.

Restricted Stone-Weierstrass Theorem

Let $\mathcal{X} \subset \mathbb{R}^d$ be compact and have at least two points. Let $\mathcal{A} \subset \mathcal{C}_1(\mathcal{X}, \mathbb{R})$ be a lattice[a] separating the points[b] of $\mathcal{X}$. Then $\mathcal{A}$ satisfies the universal approximation property for $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$.

[a] Closed under max and min.

[b] For any pair of distinct elements $x, y \in \mathcal{X}$ and real numbers $a, b \in \mathbb{R}$ with $|a - b| \leq \|y - x\|_2$, there is an $f \in \mathcal{A}$ such that $f(x) = a$ and $f(y) = b$.

Representation of piecewise affine functions

To prove the universal approximation theorem, we first show that our networks can represent all piecewise affine 1-Lipschitz functions.

Representation theorem

$\mathcal{G}_d(\mathbb{R}^d) \cap \mathcal{C}_1(\mathbb{R}^d, \mathbb{R})$ contains all the 1-Lipschitz piecewise affine functions from $\mathbb{R}^d$ to $\mathbb{R}$.

Representation of piecewise affine functions

To prove the universal approximation theorem, we first show that our networks can represent all piecewise affine 1-Lipschitz functions.

Representation theorem

$\mathcal{G}_d(\mathbb{R}^d) \cap \mathcal{C}_1(\mathbb{R}^d, \mathbb{R})$ contains all the 1-Lipschitz piecewise affine functions from $\mathbb{R}^d$ to $\mathbb{R}$.

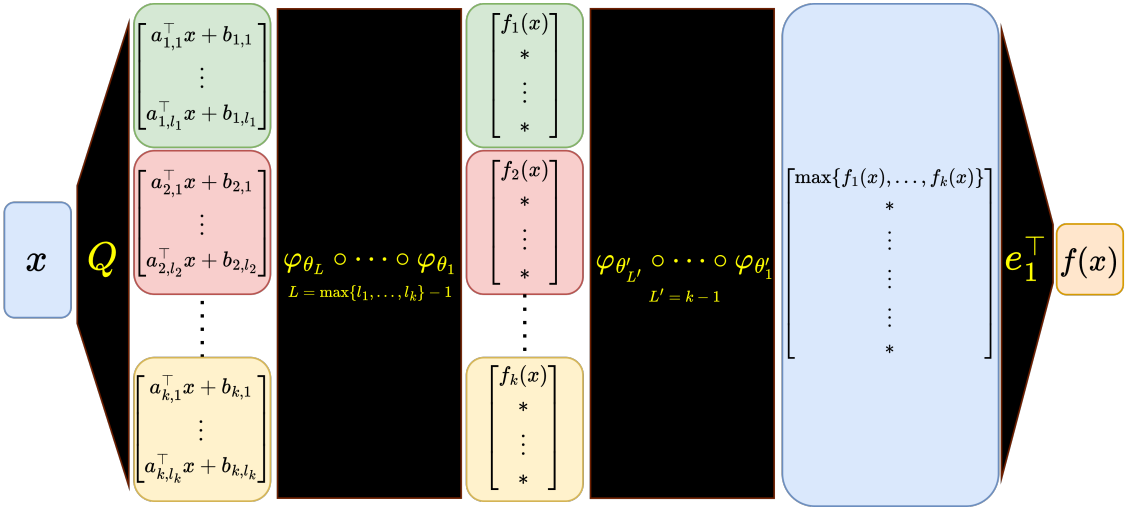
This theorem follows from the max-min representation of piecewise affine functions:

max-min representation of 1-Lipschitz piecewise affine scalar functions

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a 1-Lipschitz piecewise affine scalar function. Then, there exists a choice of scalars $b_{i,j} \in \mathbb{R}$ and vectors $a_{i,j} \in \mathbb{R}^d$, $\|a_{i,j}\|_2 \leq 1$, such that

$$f(x) = \max\{f_1(x), \dots, f_k(x)\}, \quad f_i(x) = \min\{a_{i,1}^\top x + b_{i,1}, \dots, a_{i,l_i}^\top x + b_{i,l_i}\},\ k, l_i \in \mathbb{N}.$$

Visualisation of the derivation in the proof



Preliminary properties of ReLU

Let $x, x_1, x_2 \in \mathbb{R}$. We use four of the main properties of ReLU:

- **Identity map:**

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x),$$

Preliminary properties of ReLU

Let $x, x_1, x_2 \in \mathbb{R}$. We use four of the main properties of ReLU:

- **Identity map:**

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x),$$

- **Positively homogeneous:**

$$\mathrm{ReLU}(\alpha x) = \alpha \mathrm{ReLU}(x), \text{ with } \alpha > 0,$$

Preliminary properties of ReLU

Let $x, x_1, x_2 \in \mathbb{R}$. We use four of the main properties of ReLU:

- **Identity map:**

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x),$$

- **Positively homogeneous:**

$$\mathrm{ReLU}(\alpha x) = \alpha \mathrm{ReLU}(x), \text{ with } \alpha > 0,$$

- **Representation of the maximum:**

$$\max\{x_1, x_2\} = x_1 + \mathrm{ReLU}(x_2 - x_1) = x_2 + \mathrm{ReLU}(x_1 - x_2),$$

Preliminary properties of ReLU

Let $x, x_1, x_2 \in \mathbb{R}$. We use four of the main properties of ReLU:

- **Identity map:**

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x),$$

- **Positively homogeneous:**

$$\mathrm{ReLU}(\alpha x) = \alpha \mathrm{ReLU}(x), \text{ with } \alpha > 0,$$

- **Representation of the maximum:**

$$\max\{x_1, x_2\} = x_1 + \mathrm{ReLU}(x_2 - x_1) = x_2 + \mathrm{ReLU}(x_1 - x_2),$$

- **Representation of the minimum:**

$$\min\{x_1, x_2\} = x_1 - \mathrm{ReLU}(x_1 - x_2) = x_2 - \mathrm{ReLU}(x_2 - x_1).$$

Idea of the proof of the representation theorem (part 1)

Given

$$f(x) = \max\{f_1(x), \dots, f_k(x)\}, \quad f_i(x) = \min\{a_{i,1}^\top x + b_{i,1}, \dots, a_{i,l_i}^\top x + b_{i,l_i}\},\ k, l_i \in \mathbb{N},$$

we first define the affine layer $Q : \mathbb{R}^d \to \mathbb{R}^h$ to extract all the affine pieces:

$$Q(x) = \begin{bmatrix} a_{1,1}^\top x + b_{1,1} \\ \vdots \\ a_{k,l_k}^\top x + b_{k,l_k} \end{bmatrix} \in \mathbb{R}^h,$$

with $h = l_1 + \dots + l_k$, which represents the network width.

Idea of the proof of the representation theorem (part 2)

We then extract the maxima and minima as needed via maps of the form

$$\varphi(x) = \begin{bmatrix} \max\{x_1, x_2\} \\ \min\{x_1, x_2\} \\ x_3 \\ \vdots \\ x_h \end{bmatrix} = x - 2 \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathrm{ReLU}\left(\begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 0 & \cdots & 0 \end{bmatrix} x \right),$$

which can be written as $\varphi(x) = x - \tau W^\top \mathrm{ReLU}(Wx)$, with $\tau = 2$, and

$$\mathbb{R}^{1\times h} \ni W = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 0 & \cdots & 0 \end{bmatrix}.$$

Notice that $\tau = 2/\|W\|_2^2$ since $\|W\|_2 = 1$.

Universal approximation theorem for 1-Lipschitz ResNets: Unbounded depth and fixed width

New set of networks

Fix $h \geq 3$. We now consider the set

$$\widetilde{\mathcal{G}}_{d,\sigma,h}(\mathcal{X},\mathbb{R}) := \Big\{ v^\top \circ \varphi_{\theta_L} \circ A_{L-1} \circ \cdots \circ A_1 \circ \varphi_{\theta_1} \circ Q : \mathcal{X} \to \mathbb{R} \;\Big|\; m = (1,1,1,h-3), \\ Q \in \widetilde{\mathcal{R}}_{d,m}, v \in \mathbb{R}^h, \|v\|_1 \leq 1, A_1, ..., A_{L-1} \in \widetilde{\mathcal{L}}_m, \varphi_{\theta_\ell} \in \widetilde{\mathcal{E}}_{h-3},\, L \in \mathbb{N} \Big\}.$$

$$\mathcal{L}_m = \left\{ \begin{bmatrix} A_{11} & \dots & A_{1k} \\ \vdots & \ddots & \vdots \\ A_{k1} & \dots & A_{kk} \end{bmatrix} \in \mathbb{R}^{\alpha_m \times \alpha_m} \;\middle|\; A_{ij} \in \mathbb{R}^{m_i \times m_j},\, \sum_{j=1}^{k} \|A_{ij}\|_2 \leq 1,\, i = 1, ..., k \right\},\, m \in \mathbb{N}^k,$$

$$\mathcal{R}_{d,m} = \left\{ \begin{bmatrix} B_1^\top & \cdots & B_k^\top \end{bmatrix}^\top \in \mathbb{R}^{\alpha_m \times d} \;\middle|\; B_i \in \mathbb{R}^{m_i \times d},\, \|B_i\|_2 \leq 1,\, i = 1, ..., k \right\},\, \alpha_m := \|m\|_1,$$

$$\widetilde{\mathcal{E}}_h = \Big\{ \varphi_\theta : \mathbb{R}^{h+3} \to \mathbb{R}^{h+3} \;\Big|\; \varphi_\theta(x) = \begin{bmatrix} \max\{x_1, x_2\} & \min\{x_1, x_2\} & x_3 & \widetilde{\varphi}_\theta(x_{4:})^\top \end{bmatrix},\, \widetilde{\varphi}_\theta \in \mathcal{E}_h \Big\}.$$

Theorem statement

Characterisation of the set of networks

Let $d, h \in \mathbb{N}$ with $h \geq 3$. All the functions in $\widetilde{\mathcal{G}}_{d,h}(\mathbb{R}^d, \mathbb{R})$ are 1-Lipschitz.

Representation Theorem

Any piecewise affine 1-Lipschitz function $f : \mathbb{R}^d \to \mathbb{R}$ can be represented by a network in $\widetilde{\mathcal{G}}_{d,h}(\mathbb{R}^d, \mathbb{R})$ with $h \geq d + 3$.

Universal Approximation Theorem

Let $d \in \mathbb{N}$, and $\mathcal{X} \subset \mathbb{R}^d$ be compact. The set $\widetilde{\mathcal{G}}_{d,h}(\mathcal{X}, \mathbb{R})$ satisfies the universal approximation property for $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$ if $h \geq d + 3$.

Idea of the proof for the representation theorem

Example: $f(x) = \min\{\max\{a_1^\top x, a_2^\top x\}, \max\{a_3^\top x, a_4^\top x\}\}$:

$$\mathbb{R}^d \ni x \underbrace{\mapsto}_{Q} \begin{bmatrix} a_1^\top x \\ a_2^\top x \\ 0 \\ x \end{bmatrix} \underbrace{\mapsto}_{\varphi_{\theta_1}} \begin{bmatrix} \max\{a_1^\top x, a_2^\top x\} \\ \min\{a_1^\top x, a_2^\top x\} \\ 0 \\ x \end{bmatrix} \underbrace{\mapsto}_{A_1} \begin{bmatrix} a_3^\top x \\ a_4^\top x \\ \max\{a_1^\top x, a_2^\top x\} \\ x \end{bmatrix} \underbrace{\mapsto}_{\varphi_{\theta_2}} \begin{bmatrix} \max\{a_3^\top x, a_4^\top x\} \\ \min\{a_3^\top x, a_4^\top x\} \\ \max\{a_1^\top x, a_2^\top x\} \\ x \end{bmatrix}$$

$$\underbrace{\mapsto}_{A_2} \begin{bmatrix} \max\{a_1^\top x, a_2^\top x\} \\ \max\{a_3^\top x, a_4^\top x\} \\ 0 \\ x \end{bmatrix} \underbrace{\mapsto}_{\varphi_{\theta_3}} \begin{bmatrix} \max\{a_1^\top x, a_2^\top x, a_3^\top x, a_4^\top x\} \\ \min\{\max\{a_1^\top x, a_2^\top x\}, \max\{a_3^\top x, a_4^\top x\}\} \\ 0 \\ x \end{bmatrix} \underbrace{\mapsto}_{e_2^\top} f(x).$$

Final remarks

- Since $\widetilde{\mathcal{E}}_h \subset \mathcal{E}_{h+3}$, the same universality result applies by replacing the highly-structured gradient modules considered in $\widetilde{\mathcal{G}}_{d,d+3}(\mathcal{X}, \mathbb{R})$ with the unconstrained ones in $\mathcal{E}_{h+3}$ and intersecting with $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$.

Final remarks

- Since $\widetilde{\mathcal{E}}_h \subset \mathcal{E}_{h+3}$, the same universality result applies by replacing the highly-structured gradient modules considered in $\widetilde{\mathcal{G}}_{d,d+3}(\mathcal{X}, \mathbb{R})$ with the unconstrained ones in $\mathcal{E}_{h+3}$ and intersecting with $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$.
- To obtain universality while maintaining the width fixed, we introduced affine layers between the residual gradient steps. Setting them to identity maps would lead to a subset of $\widetilde{\mathcal{G}}_{d,h}(\mathcal{X}, \mathbb{R})$ and of $\mathcal{G}_d(\mathcal{X}, \mathbb{R})$ which might not be universal.

Final remarks

- Since $\widetilde{\mathcal{E}}_h \subset \mathcal{E}_{h+3}$, the same universality result applies by replacing the highly-structured gradient modules considered in $\widetilde{\mathcal{G}}_{d,d+3}(\mathcal{X}, \mathbb{R})$ with the unconstrained ones in $\mathcal{E}_{h+3}$ and intersecting with $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$.
- To obtain universality while maintaining the width fixed, we introduced affine layers between the residual gradient steps. Setting them to identity maps would lead to a subset of $\widetilde{\mathcal{G}}_{d,h}(\mathcal{X}, \mathbb{R})$ and of $\mathcal{G}_d(\mathcal{X}, \mathbb{R})$ which might not be universal.
- Our proofs strongly rely on the properties of ReLU. The same results could be obtained by any other activation functions that are positively homogeneous, can represent the identity map, and the entrywise maximum and minimum functions.

Final remarks

- Since $\widetilde{\mathcal{E}}_h \subset \mathcal{E}_{h+3}$, the same universality result applies by replacing the highly-structured gradient modules considered in $\widetilde{\mathcal{G}}_{d,d+3}(\mathcal{X}, \mathbb{R})$ with the unconstrained ones in $\mathcal{E}_{h+3}$ and intersecting with $\mathcal{C}_1(\mathcal{X}, \mathbb{R})$.
- To obtain universality while maintaining the width fixed, we introduced affine layers between the residual gradient steps. Setting them to identity maps would lead to a subset of $\widetilde{\mathcal{G}}_{d,h}(\mathcal{X}, \mathbb{R})$ and of $\mathcal{G}_d(\mathcal{X}, \mathbb{R})$ which might not be universal.
- Our proofs strongly rely on the properties of ReLU. The same results could be obtained by any other activation functions that are positively homogeneous, can represent the identity map, and the entrywise maximum and minimum functions.
- The elements of $\mathcal{G}_d(\mathcal{X}, \mathbb{R})$ and $\widetilde{\mathcal{G}}_{d,h}(\mathcal{X}, \mathbb{R})$ are neural networks that can be numerically implemented. All the weight constraints can be efficiently enforced using projected gradient descent. Spectral norms can be estimated using the power method.

THANK YOU FOR THE ATTENTION

davidemurari.com/onelip to read the paper

APPENDIX

1-Lipschitz Networks for Inverse Problems

The Proximal Gradient Descent Method

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) + \gamma g(\mathbf{x})\right), \quad f : \mathbb{R}^d \to \mathbb{R},\ g : \mathbb{R}^d \to \mathbb{R} \cup \{\pm\infty\}, \tag{1}$$

where f is a data-fidelity term, g is a regularisation term, and $\gamma > 0$.

The Proximal Gradient Descent Method

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) + \gamma g(\mathbf{x})\right), \ f : \mathbb{R}^d \to \mathbb{R}, \ g : \mathbb{R}^d \to \mathbb{R} \cup \{\pm\infty\}, \tag{1}$$

where f is a data-fidelity term, g is a regularisation term, and $\gamma > 0$.

Example:

$$f(\mathbf{x}) = \frac{1}{2}\|K\mathbf{x} - \mathbf{y}\|_2^2, \ g(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2 \quad \text{(Ridge Regression)}.$$

The Proximal Gradient Descent Method

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) + \gamma g(\mathbf{x})\right), \ f : \mathbb{R}^d \to \mathbb{R}, \ g : \mathbb{R}^d \to \mathbb{R} \cup \{\pm\infty\}, \tag{1}$$

where f is a data-fidelity term, g is a regularisation term, and $\gamma > 0$.

Example:

$$f(\mathbf{x}) = \frac{1}{2}\|K\mathbf{x} - \mathbf{y}\|_2^2, \ g(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2 \quad \text{(Ridge Regression)}.$$

Assume $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R} \cup \{\pm\infty\}$ convex, f continuously differentiable, g continuous and proper. A method to solve (1) is the Proximal Gradient Descent Method:

$$\mathbf{x}_{k+1} = \operatorname{prox}_{\gamma g,\tau}\left(\mathbf{x}_k - \tau\nabla f(\mathbf{x}_k)\right), \ \tau > 0,$$

$$\operatorname{prox}_{\gamma g,\tau}(\mathbf{x}) = \operatorname*{arg\,min}_{\mathbf{z}\in\mathbb{R}^d} \left(\frac{1}{2\tau}\|\mathbf{x} - \mathbf{z}\|_2^2 + \gamma g(\mathbf{z})\right).$$

Example: Projected Gradient Descent

$\Omega \subset \mathbb{R}^d$ non-empty, closed, convex set. $f : \mathbb{R}^d \to \mathbb{R}$ convex and continuously differentiable.

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \iff \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) + i_\Omega(\mathbf{x}), \; i_\Omega(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in \Omega, \\ +\infty, & \mathbf{x} \notin \Omega. \end{cases}$$

Example: Projected Gradient Descent

$\Omega \subset \mathbb{R}^d$ non-empty, closed, convex set. $f : \mathbb{R}^d \to \mathbb{R}$ convex and continuously differentiable.

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \iff \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) + i_\Omega(\mathbf{x}), \; i_\Omega(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in \Omega, \\ +\infty, & \mathbf{x} \notin \Omega. \end{cases}$$

Here, we have that if $i_\Omega =: g$, the proximal operator is an orthogonal projection operator:

$$\mathrm{prox}_{\gamma g,\tau}(\mathbf{x}) = \arg\min_{\mathbf{z}\in\mathbb{R}^d} \left(\frac{1}{2\tau}\|\mathbf{x}-\mathbf{z}\|_2^2 + \gamma i_\Omega(\mathbf{z}) \right) = \arg\min_{\mathbf{z}\in\Omega} \|\mathbf{x}-\mathbf{z}\|_2^2 = \mathrm{proj}_\Omega(\mathbf{x}).$$

Example: Projected Gradient Descent

$\Omega \subset \mathbb{R}^d$ non-empty, closed, convex set. $f : \mathbb{R}^d \to \mathbb{R}$ convex and continuously differentiable.

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}) \iff \min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) + i_\Omega(\mathbf{x}), \; i_\Omega(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in \Omega, \\ +\infty, & \mathbf{x} \notin \Omega. \end{cases}$$

Here, we have that if $i_\Omega =: g$, the proximal operator is an orthogonal projection operator:

$$\mathrm{prox}_{\gamma g,\tau}(\mathbf{x}) = \arg\min_{\mathbf{z}\in\mathbb{R}^d} \left(\frac{1}{2\tau}\|\mathbf{x}-\mathbf{z}\|_2^2 + \gamma i_\Omega(\mathbf{z}) \right) = \arg\min_{\mathbf{z}\in\Omega} \|\mathbf{x}-\mathbf{z}\|_2^2 = \mathrm{proj}_\Omega(\mathbf{x}).$$

The proximal gradient method then becomes the projected gradient descent method:

$$\mathbf{x}_{k+1} = \mathrm{proj}_\Omega(\mathbf{x}_k - \tau \nabla f(\mathbf{x}_k)).$$

Example: ISTA (cfr. SINDy)

Let $f(\mathbf{x}) = \frac{1}{2}\|K\mathbf{x} - \mathbf{y}\|_2^2$, $g(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_{i=1}^{d} |x_i|$, and $\gamma > 0$ the regularisation parameter.

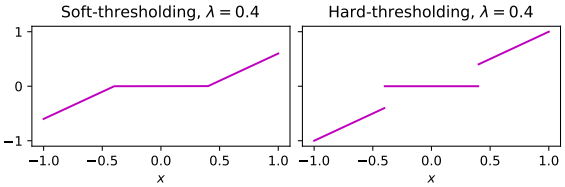
Example: ISTA (cfr. SINDy)

Let $f(\mathbf{x}) = \frac{1}{2}\|K\mathbf{x} - \mathbf{y}\|_2^2$, $g(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_{i=1}^d |x_i|$, and $\gamma > 0$ the regularisation parameter.

The Proximal Gradient Descent then writes

$$\mathbf{x}_{k+1} = \operatorname{prox}_{\gamma g, \tau}\left(\mathbf{x}_k - \tau K^\top (K\mathbf{x} - \mathbf{y})\right) = S_{\tau\gamma}\left(\mathbf{x}_k - \tau K^\top (K\mathbf{x} - \mathbf{y})\right),$$

$$(S_\lambda(\mathbf{x}))_i = \begin{cases} x_i - \lambda, & x_i > \lambda, \\ 0, & |x_i| \leq \lambda, \\ x_i + \lambda, & x_i < -\lambda, \end{cases} \quad \lambda > 0,\ i = 1, \ldots, d.$$



The Plug-and-Play Method

There are two problems with what we saw in the two previous slides:

1. It is extremely hard to define a **good regulariser** for any given task,
2. The **proximal operator** of a generic regulariser g is **not easy to compute**.

The Plug-and-Play Method

There are two problems with what we saw in the two previous slides:

1. It is extremely hard to define a **good regulariser** for any given task,
2. The **proximal operator** of a generic regulariser g is **not easy to compute**.

Solution: The Plug-and-Play method is defined by replacing $\mathrm{prox}_{\gamma g,\alpha}$ with a Neural Network:

$$\text{Plug-and-Play:} \quad \mathbf{x}_{k+1} = \mathcal{N}_\theta(\mathbf{x}_k - \tau \nabla f(\mathbf{x}_k)), \ \mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^d. \tag{2}$$

The network $\mathcal{N}_\theta$ is typically trained offline to denoise images:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{N}_\theta(\mathbf{x}_i + \delta_i) - \mathbf{x}_i\|_2^2, \quad \delta_1, ..., \delta_N \sim \mathcal{D}.$$

The Plug-and-Play Method

There are two problems with what we saw in the two previous slides:

1. It is extremely hard to define a **good regulariser** for any given task,
2. The **proximal operator** of a generic regulariser g is **not easy to compute**.

Solution: The Plug-and-Play method is defined by replacing $\mathrm{prox}_{\gamma g,\alpha}$ with a Neural Network:

$$\text{Plug-and-Play:} \quad \mathbf{x}_{k+1} = \mathcal{N}_\theta(\mathbf{x}_k - \tau \nabla f(\mathbf{x}_k)), \ \mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^d. \tag{2}$$

The network $\mathcal{N}_\theta$ is typically trained offline to denoise images:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{N}_\theta(\mathbf{x}_i + \delta_i) - \mathbf{x}_i\|_2^2, \quad \delta_1, ..., \delta_N \sim \mathcal{D}.$$

Convergence guarantees

Assume f is μ-strongly convex, L-smooth, and $\tau \in (0, 2/L)$. Then if $\mathcal{N}_\theta$ is 1-Lipschitz, the iterates in (2) converge to a unique fixed point.

Averaged maps

α-averaged map

The map $T : \mathbb{R}^d \to \mathbb{R}^d$ is averaged if there exists $\alpha \in (0, 1)$ and a 1-Lipschitz map $F : \mathbb{R}^d \to \mathbb{R}^d$ such that $T = (1 - \alpha)\mathrm{id} + \alpha F$. The composition of averaged maps is again averaged. Patrick L Combettes and Isao Yamada. "Compositions and Convex Combinations of Averaged Nonexpansive Operators". In: *Journal of Mathematical Analysis and Applications* 425.1 (2015), pp. 55–70, Proposition 2.4

Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex, continuously-differentiable, and L-smooth. Then if $\tau \in (0, 2/L)$ the map $T(\mathbf{x}) = \mathbf{x} - \tau \nabla f(\mathbf{x})$ is averaged with $\alpha = \tau L/2$.

Convergence under convexity

Convergence Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable, convex, and L-smooth. Assume $\tau \in (0, 2/L)$. Then $G = \mathrm{id} - \tau \nabla f$ is $\tau L/2$ averaged. Further assume that $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is averaged. Let $T = \mathcal{N}_\theta \circ G$. Assuming that $\mathrm{Fix}(T) \neq \emptyset$, the Plug-and-Play iterates $\mathbf{x}_{k+1} = T(\mathbf{x}_k)$ will converge to a fixed point.

Convergence under convexity

Convergence Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable, convex, and L-smooth. Assume $\tau \in (0, 2/L)$. Then $G = \mathrm{id} - \tau \nabla f$ is $\tau L/2$ averaged. Further assume that $\mathcal{N}_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is averaged. Let $T = \mathcal{N}_\theta \circ G$. Assuming that $\mathrm{Fix}(T) \neq \emptyset$, the Plug-and-Play iterates $\mathbf{x}_{k+1} = T(\mathbf{x}_k)$ will converge to a fixed point.

Our networks are explicit Euler steps for the gradient of $f(x) = 1^\top \mathrm{ReLU}^2(Ax + b)/2$, which is convex and its gradient is

$$\nabla f(x) = A^\top \mathrm{ReLU}(Ax + b),$$

which is $\|A\|_2^2$-Lipschitz. This means that the layers of our 1-Lipschitz network are averaged if $0 < \tau_i < 2/\|A_i\|_2^2$, and hence so is the full network $\mathcal{N}_\theta$.

The Network $\mathcal{N}_\theta$ Trained as Denoiser

PSNR (Peak Signal-to-Noise Ratio)

$$\mathrm{PSNR}(\hat{\mathbf{x}}, \mathbf{x}^*) = 10 \log_{10} \left(\frac{\max_{i,j,k} |x^*_{i,j,k}|^2}{\frac{1}{3 \cdot 321 \cdot 481} \sum_{i,j,k} |x^*_{i,j,k} - \hat{x}_{i,j,k}|^2} \right).$$

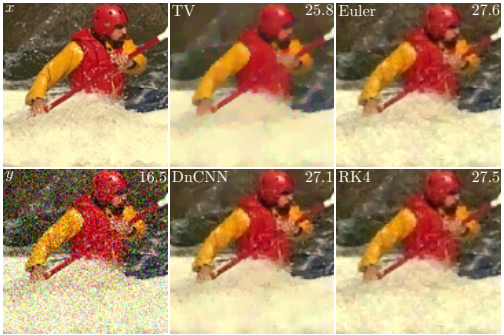


Figure 2: Image from BSDS500 dataset, composed of 500 natural colour images of size 321×481.

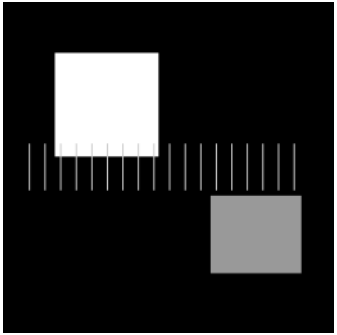
What do we mean with Deblurring?

- Let us consider the **inverse problem of deblurring**: we assume that we are given measurements $y = Kx + \varepsilon$, where $K\mathbf{x} = k * \mathbf{x}$ is a convolution operation representing a motion blur.
- The ill-posedness of this problem is manifested in the instability of the inverse of the convolution; as a consequence of this, a naive inversion of the measurements will blow up the noise in the measurements.
- The data-fidelity term is

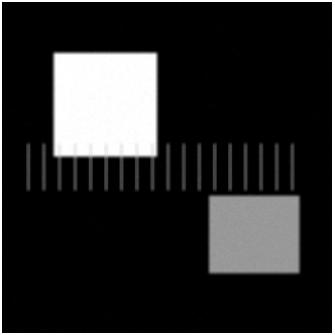
$$f(\mathbf{x}) = \frac{1}{2}\|K\mathbf{x} - \mathbf{y}\|_2^2.$$

Visualisation of the ill-posedness

Original, x

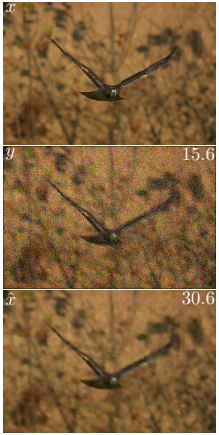


Blur + noise, $y = Kx + \varepsilon$



Naive inverse $K^{-1}y = \hat{x}$

Use in a Deblurring Task



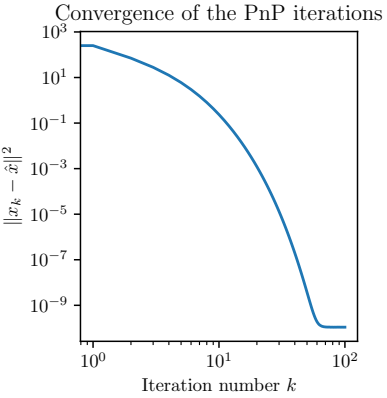


Figure 3: Using the learned Euler denoiser to solve an ill-posed inverse problem (deblurring) in a PnP fashion, with convergence guarantee.