École polytechnique fédérale de Lausanne

Master thesis spring 2024

Master in Applied Mathematics

Infimal convolution of Total Variation and Convex Neural Network regularizer

Author:
Wael Gafaiti

Supervisor:
Professor Marta M. Betcke

Abstract

In image-reconstruction inverse problem framework, regularizers of any kind have arised in the literature. We propose here a study of convex infimal convolution regularizers between the Total Variation regularizer (TV) and a convex ridge regularizer neural network (CRRNN). The latter is a deep learning based regularizer which disposes of theoretical guarantees over the reconstruction. We also trained a CRRNN model under the convex infimal convolution framework using implicit layers and t-unfolding Alternating Direction Method of Multipliers (ADMM) algorithm. Through numerical experiments, we compare performances of implicit layers and t-unfolding gradient descent on CRRNN models, two different training procedures. Then, we study performances of the convex infimal convolution regularizer using pretrained and trained under the convexinfimal convolution framework CRRNN models. We finally relax the convexity assumption of our infimal convolution regularizers by considering a weakly convex ridge regularizer neural network (WCRRNN) instead of the CRRNN and a strongly convex approximation of the TV regularizer instead of the standard TV regularizer.

Contents

1 Introduction

1.1 Background

We focus on regularizers used in the specific inverse problem of image reconstruction. To give some context, we suppose that we observe:

$$y = H(x) + n \in \mathbb{R}^{p\times q}, \tag{1}$$

where $H(\cdot) : \mathbb{R}^{n\times m} \to \mathbb{R}^{p\times q}$ is a linear operator over $\mathbb{R}^{n\times m}$, $x \in \mathbb{R}^{n\times m}$ and $n \in \mathbb{R}^{p\times q}$. In fact, $H(x)$ corresponds to series of linear measurements applied to the discrete representation of the image x. And, n is a (Gaussian) additive noise related to the measurement process. The goal is to reconstruct x given y. Due to the presence of the noise and the fact that H could be ill-conditioned (especially if $pq < nm$), the reconstruction can result in an undetermined problem. Hence, to solve (1) in such conditions, we often compute:

$$x^* \in \underset{x\in\mathbb{R}^{n\times m}}{\arg\min} ||H(x) - y||_2^2 + R(x), \tag{2}$$

where $R : \mathbb{R}^{n\times m} \to \mathbb{R}$ is a regularizer. The role of the latter is to include all the prior information about x to counteract the ill-posedness of (1). Many regularizers and approaches have been studied in the literature such as the Tikhonov regularizer and the total variation (TV). More recently, learnable regularizer through convex ridge regularizer neural network (CRRNN) and a weakly convex ridge regularizer neural network (WCRRNN) have been presented and studied in [10] and [11]. Despite the deep learning framework, theoretical guarantees are drawn over the reconstruction, making those neural network regularizers very interesting to use in medical fields for example, where reliability and interpretability over the reconstruction methods are convenient. In fact, many deep-learning-based regularizer have emerged in the last decade [14]. They show very good results, however, as many deep-learning model, it is very hard to understand their structure. This is one of the fields of study of AI safety. Hopefully, CRRNN and WCRRNN are built to be interpretable.
When $H = Id$, where Id is the identity operator over $\mathbb{R}^{n\times m}$, we refer to the denoising task as illustrated on figure (1) and we will mainly focus on this framework since the training of the CRRNN and the WCRRNN models is done under denoising only. In fact, the denoising task will also appear in (2) for any arbitrary linear operators H because of the Alternating Direction Method of Multipliers (ADMM) scheme or the Forward/Backward splitting, two common methods to solve this inverse problem.

The linear measurements H can take different shapes w.r.t. the experiment applied such a *Magnetic Resonance Imaging (MRI)* or *Computed Tomography (CT)* to mention just a few [3].

Figure 1: Example of denoising task on "Lenna picture". *Top left:* Clean image. *Top right:* A noise of mean $\mu = 0$ and a noise level $\sigma = \frac{25}{255}$ has been added. *Bottom left:* Image denoised though the TV regularizer. *Bottom right:* Image denoised through the CRRNN regularizer.

1.2 Motivation

This paper is about the study of the convex infimal convolution of the TV regularizer and CRRNN models. In fact, the TV regularizer is a very popular regularizer, however, it presents some weaknesses as illustrated in Figure (2).

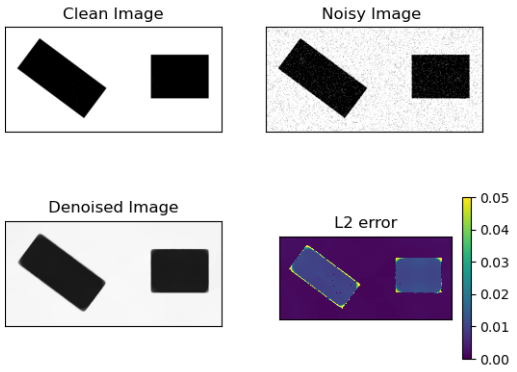

Figure 2: There are several variants of the discrete total variation. The most common one is the isotropic total variation based on the Euclidean norm which tends to round the edges in reconstruction as we can observe it on the figure.

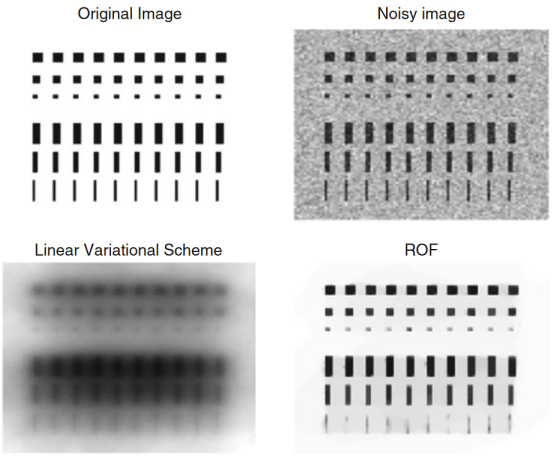

Figure 3: The bottom right image corresponds to the reconstruction through the TV regularizer. And, we see that the reconstruction seems performs better on blocky images. This figure is from (Section 2.3, [6]).

Hence, we hope that the convex infimal convolution will counteract its weaknesses by still preserving the strengths. Thus, the choice of using a CRRNN model makes sense since it is a learnable and can fullfill some specific tasks, such as counteract the weaknesses of the TV regularizer. We could choose any learnable regularizer like the many others that exist. However, CRRNN models offer us theoretical properties such as the stability of the reconstruction and the existence of a minimum which are preserved by infimal convolution in our framework. This is not the case of other deep learning based regularizers. We could also see this convex infimal convolution in the other way by trying to counteract the weaknesses of the CRRNN models with the TV regularizer.

Why should we also consider weakly convex regularizers? It has been shown so far that relaxing the assumption of convexity property of regularizers to weakly convexity property can improve significantly the performances in the denoising task and still preserving the theoretical guarantees of the reconstruction. In some cases, weakly convex regularizers are more interpretable than the convex ones as stated in [17]. Hence, we construct a weakly convex infimal convolution regularizer by considering weakly convex neural network models and a strongly convex variant of the TV regularizer.

1.3 Notations

All along this paper, the following notation will be used;

- R_{TV} refers to the TV regularizer. We index by R^{μ}_{TV} when we refer to the strong convex approximation of the TV regularizer (see section 5.1).
- R_θ refers to ridge regularizer neural network with θ corresponding to the trained parameters of the neural network. We index by C to get R^C_θ if we need to specify that we are considering a CRRNN model and index by W to get R^W_θ if we need to specify that we are considering a WCRRNN model.
- Ssim corresponds to the *Structural Similarity Index Measure*which allows us to a metric between two images. Higher the ssim score is, closer are two images. The score of ssim is between 0 and 1.
- Psnr corresponds to the *Peak Signal to Noise Ratio* which plays the same role as ssim. Higher the psnr score is, closer are two images. The score of psnr is non-negative.
- We say that a function $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ is weakly convex if there exists $C > 0$ s.t. f is C-weakly convex (see definition 5.2).

1.4 Main results

In this work, we focus on the following regularizer

$$R^{infConv}_{\alpha,\beta,\theta}(x) = \min_{y\in\mathbb{R}^{n\times m}} \left(\alpha R_{TV}(x-y) + \beta R_\theta(y)\right) \tag{3}$$

In the convex case (i.e. $R_\theta = R^C_\theta$), we explain how to solve the inverse problem related to this regularizer via the ADMM algorithm. Then, we compare two techniques of training neural network based models: implicit layers and t-unfolding of the gradient descent algorithm. In fact, the both approaches will be needed and used to train a CRRNN model under the convex infimal convolution framework. We studied then the performances of $R^{infConv}_{\alpha,\beta,\theta}$ using CRRNN models non trained under the convex infimal convolution framework (we refer to such CRRNN models as pretrained CRRNN models). Then, we explain how to train a CRRNN model under this convex infimal convolution framework and study its performances.

Finally, we move to the weakly convex case. We explore and show ways to construct a weakly convex infimal convolution regularizer. In fact, we consider WCRRNN model instead of the CRRNN and a strongly convex approximation of the TV regularizer instead of the standard TV regularizer and study its numerical performances.

2 Infimal Convolution Regularizer: Convex case

The new regularizer (3) is an infimal convolution of $f(x) = \alpha R_{TV}(x)$ and $g(x) = \beta R^C_\theta(x)$ which is formally defined as:

$$(f\square g)(x) = \min_{w\in\mathbb{R}^{n\times m}} (f(x-w) + g(w)) \tag{4}$$

$$= \min_{w\in\mathbb{R}^{n\times m}} (\alpha\, R_{TV}(x-w) + \beta R^C_\theta(w)). \tag{5}$$

We state here some basic properties of the infimal convolution (see [1] for more details), which are useful to get acquainted with especially in our context, and the definition of convexity:

Proposition 2.1. *Let $f, g : \mathbb{R}^{n\times m} \to \mathbb{R}$ be some arbitrary real functions, then:*

1. *$f\square g = g\square f$,*
2. *$dom(f\square g) = dom(f) + dom(g)$, where $dom(f) := \{x \in \mathbb{R}^{n\times m} | f(x) < \infty\}$,*
3. *if f and g are convex functions, then $f\square g$ is also a convex function.*

Definition 2.1. *We say that a function $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ is called convex if and only if it satisfies the following equation for all $x, y \in \mathbb{R}^{n\times m}$ and $\lambda \in [0, 1]$:*

$$\lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y). \tag{6}$$

By property (3), this regularizer is also a convex function assuming $\alpha, \beta > 0$, $\alpha R_{TV}(\cdot)$ and $\beta R^C_\theta(\cdot)$ are convex functions since $R_{TV}(\dot{)}$ and $R^C_\theta(\cdot)$ are convex as discussed in sections 2.1 and 2.2. This is an important property as it allows us to use some specific algorithms such as gradient based algorithms to solve the related inverse problem.

2.1 Total Variation

The Total Variation regularizer has been first introduced by Rudin, Osher and Fatemi in [16] with the spirit of introducing a regularizer which ensures spatial regularity while preserving the edges. These authors studied the regularizer in the continuous setting for $u \in L^1(\Omega)$, where $\Omega \subset \mathbb{R}^2$. In fact, $u(x, y)$ corresponds to the grey-level value of the image at coordinates $(x, y) \in \Omega$. The total variation in this setting is read as:

$$J(u) \coloneqq \sup\Big\{ -\int_\Omega u div\varphi \, dx : \varphi \in C^\infty_c(\Omega, \mathbb{R}^2), \|\varphi\|_\infty \leq 1 \Big\}. \tag{7}$$

We note that if $u \in W^{1,1}(\Omega)$, then, thanks to the integration by parts formula and the fact that all $\varphi \in C^\infty_c(\Omega, \mathbb{R}^2)$ vanish in the boundary $\delta\Omega$, we have:

$$-\int_\Omega u div\varphi \, dx = \int_\Omega \varphi \cdot \nabla u \, dx,$$

$\forall \varphi \in C^\infty_c(\Omega, \mathbb{R}^2)$. Hence, by taking the supremum over $\varphi \in C^\infty_c(\Omega, \mathbb{R}^2)$ such that $\|\varphi\|_\infty \leq 1$, we obtain:

$$J(u) = \int_\Omega |\nabla u| \, dx. \tag{8}$$

From (8), the TV in the discrete setting for $u \in \mathbb{R}^{n \times m}$ is defined as follows:

$$J(u) = \sum_{\substack{1 \leq i \leq n, \\ 1 \leq j \leq m}} \sqrt{(\nabla u)^2_{i,j,1} + (\nabla u)^2_{i,j,2}}, \tag{9}$$

$$= \|\nabla u\|_1 \tag{10}$$

where $\nabla \in \mathbb{R}^{n \times m} \times \mathbb{R}^{n \times m}$ and is defined s.t.:

$$(\nabla u)_{i,j,1} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } 1 \leq i < n \\ 0 & \text{otherwise} \end{cases}$$

$$(\nabla u)_{i,j,2} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } 1 \leq j < m \\ 0 & \text{otherwise.} \end{cases}$$

This definition of the discrete TV usually refers to the isotropic TV. In continuous and discrete settings, the TV is a convex function in the variable u and it has been shown theoretically that this discrete approximation is consistent with the TV regularizer defined in the continuous setting (*Proposition 3.1*, [9]). Hence, the norm is defined in the following way $R_{TV}(\cdot) \coloneqq J(\cdot)$. Thus, we understand of why the TV regularizer has such weaknesses mentioned in section 1.2. The reconstructed image can be "cartoonish" if the corresponding hyperparameter λ is too large. More details concerning the weaknesses of the TV are given in [6]. Usually, we set λ to $5 \cdot 10^{-2}$ when the standard deviation of the noise is under 25% of the maximum value taken of the discrete representation of the image.

2.1.1 Reconstruction method

In the TV framework, the reconstructed image x^* corresponds to the solution of the following inverse problem:

$$x^* = \underset{x \in \mathbb{R}^{n \times m}}{\arg\min} \frac{1}{2} \|y - H(x)\|^2_2 + \lambda R_{TV}(x). \tag{11}$$

Numerically, the reconstruction x^* can be recovered by using gradient-based methods. One very popular used algorithm in this framework is the Fast Iterative Shrinkage Tresholding algorithm (FISTA) 1 described in [2].

Algorithm 1 FISTA

Require: $L \geq L(f)$, where $L(f)$ is the Lipschitz constant of ∇f, $y_1 = x_0 \in \mathbb{R}^d$ and $t_1 = 1$.
while tolerance not reached **do**
$x_k = p_L(y_k)$
$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
$y_{k+1} = x_k + (\frac{t_k - 1}{t_{k+1}})(x_k - x_{k-1})$,
end while

where $f(x) = \|H(x) - y\|_2^2$ and $p_L(y) = \arg\min_{x \in \mathbb{R}^{n \times m}} \left\{ \frac{L}{2} \|x - (y - \frac{1}{L}\nabla f(y))\|_2^2 + R_{TV}(x) \right\}$. The implementation of the proximal map of Moreau$p_L(\cdot)$ is more detailed in [8], [2].

However, we note that in the denoising task, that we can use only the proximal map of Moreau $p_L(\cdot)$ by choosing L and $f(\cdot)$ in an adequate manner. In fact, by choosing $f(\cdot)$ s.t. $\nabla f(\cdot) = 0$ and $L = \frac{1}{\lambda}$, it will solve (11).

2.2 Convex ridge regularizer neural network

CRRNN is a neural network based regularizer suggested and studied in [10]. Contrarily to the TV regularizer, the CRRNN is not directly based on the spatial gradient of the discrete representation of images. In fact, the regularity of an image $x \in \mathbb{R}^{n \times m}$ is measured as follows:

$$R_\theta^C : x \mapsto \sum_{i=1}^{N_C} \sum_{k \in \mathbb{Z}^2} \psi_i((h_i * x)[k]), \tag{12}$$

where $\psi_i : \mathbb{R} \to \mathbb{R}$ are convex learnable profile functions, h_i is an impulse response of a 2D convolutional filter, $(h_i * x)[k]$ is the the value of the k-th pixel of the filtered image $h_i * x$ and N_C is the number of channels in the context of convolutional channels. We use the notation $R_\theta^C(\cdot)$ to underline the dependencies of the regularizer to its learnable parameters θ. (12) can be reformulated in a more generic form:

$$R_\theta^C : x \mapsto \sum_i \psi_i(w_i^T \mathbf{x}), \tag{13}$$

where ψ_i are also convex learnable profile functions, $w_i \in \mathbb{R}^{nm}$ are the learnable weights and $\mathbf{x} \in \mathbb{R}^{nm}$ is the flatten version of the image $x \in \mathbb{R}^{n \times m}$. The formulation as in (13) is preferred since it simplifies the notations and makes the further computations clearer. By constraining the profile functions ψ_i to be convex, it ensures that the regularizer $R_\theta^C(\cdot)$ is convex. Furthermore, the derivatives of the profiles ψ_i are constrained to be Lipschitz continuous, i.e. $\psi_i \in C^{1,1}(\mathbb{R})$.

2.2.1 Training

We briefly explain how the CRRNN models are trained since it will be needed to train CRRNN models under the infimal convolution framework. More details on the training can be found in (Sections II B. and IV, [10]). The regularizer $R_\theta^C(\cdot)$ is trained under the denoising task, i.e $H(\cdot) = Id$ s.t. the solution of the following inverse problem is close to the original image:

$$x^* = \arg\min_{x \in \mathbb{R}^d} \frac{1}{2}\|x - y\|_2^2 + \lambda R_\theta^C(x), \tag{14}$$

where λ is an hyperparameter to tune. The objective function to minimize is differentiable. Hence, the denoised image x^* can be interpreted as the unique fixed point of:

$$T_{R_\theta, \alpha, \lambda} : \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times m} \tag{15}$$

$$x \mapsto x - \alpha((x - y) + \lambda \nabla R_\theta^C(x)). \tag{16}$$

The iterations of the above operator lead to the implementation of a gradient descent with step size α, which converges if $\alpha \in (0, \frac{2}{1+\lambda L_\theta})$, where L_θ is the Lipschitz constant of $\nabla R_\theta^C(\cdot)$. In fact, $\nabla R_\theta^C(\cdot)$ is learned with a bound $L \geq L_\theta$ instead of $R_\theta^C(\cdot)$*(Proposition 4.1, [10])*. Then, we can reconstruct the regularizer cost function

$R_\theta^C(\cdot)$ uniquely up to an additive constant. Considering $R_\theta^C(\cdot)$ under its generic form (13) allows us to compute the gradient $\nabla R_\theta^C(\cdot)$ as follows:

$$\begin{aligned}\frac{\delta}{\delta \mathbf{x}_i} R_\theta^C(x) &= \frac{\delta}{\delta \mathbf{x}_i} \sum_j \psi_j(\mathbf{w_j}^T \mathbf{x}) \\ &= \sum_j \mathbf{w}_{j,i} \psi'_j(\mathbf{w_j}^T \mathbf{x}).\end{aligned}$$

Then, the gradient w.r.t. to $\mathbf{x}$ is

$$\nabla R_\theta^C(x) = \mathbf{W}^T \sigma(\mathbf{W}\mathbf{x}), \tag{17}$$

where $\mathbf{W} = [\mathbf{w_1} \cdots \mathbf{w_c}]^T \in \mathbb{R}^{c \times nm}$ and $\sigma(\mathbf{W}\mathbf{x}) = [\psi_1((\mathbf{W}\mathbf{x})_1)' \cdots \psi_c((\mathbf{W}\mathbf{x})_c)']^T$ is a pointwise activation function. The components of σ are Lipschitz continuous, because of the restriction we set on the derivatives of the learnable profiles, and increasing since the profiles are convex. Now, the gradient step operator becomes:

$$T_{R_\theta^C,\lambda,\alpha}(x) = (1-\alpha)x + \alpha(y - \lambda \mathbf{W}^T \sigma(\mathbf{W}\mathbf{x})). \tag{18}$$

Hence, the operator (18) corresponds to a one-hidden-layer convolutional neural network with a bias and a skip connection.

Let's have a few words over the implementation of the training now. To do so, let $\{x_i\}_{i=1}^I$ be the set of original images and $\{y_i\}_{i=1}^I = \{x_i + n_i\}_{i=1}^I$ the set of the corresponding noisy images, where n_i is the realisation of a noise. Hence, we are looking for θ^*, λ^* s.t, for any loss function $\mathcal{L} : \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$.

$$\theta^*, \lambda^* \in \arg\min_{\theta,\lambda} \sum_{i=1}^I \mathcal{L}(x_i, T^t_{R_\theta^C,\lambda,\alpha}(y_i)), \tag{19}$$

where $T^t_{R_\theta^C,\lambda,\alpha}$ is the *t-fold* composition of $T_{R_\theta^C,\lambda,\alpha}$, i.e. $T^t_{R_\theta^C,\lambda,\alpha} := T_{R_\theta^C,\lambda,\alpha} \circ \cdots \circ T_{R_\theta^C,\lambda,\alpha}$. Ideally, we would like to solve problem (19) with $t = \infty$ using implicit layers or bi-level optimization scheme, however the authors claimed that this is unnecessary to fully solve in this way. In fact, here $T^\infty_{R_\theta^C,\lambda,\alpha}$ is approximated by $T^t_{R_\theta^C,\lambda,\alpha}$. Hence, the *t-step denoiser NN* $T_{R_\theta^C,\lambda,\alpha}$ is trained s.t.

$$T^t_{R_\theta^C,\lambda,\alpha}(y_i) \simeq x_i, \tag{20}$$

for all $i = 1, \cdots, I$. The authors named this as the unfolding of the gradient-descent algorithm for t iterations with shared parameters across iterations. It reduces drastically the training time compared to implicit layers approach (more details in section 3).

The parameters which have to be trained are $\lambda \in \mathbb{R}, w_1, \cdots, w_c \in \mathbb{R}^{nm}$ and the parameter related to the linear-splines σ_i. The constraints needed on $\sigma_i = \psi'_i$ are:

1. They have to be increasing,
2. They have to take the value 0 at least at one point, to ensure the existence of a minimum of their integrate ψ_i.

Hence, the following implementation is proposed. The learnable $\sigma_{c_i} : \mathbb{R} \to \mathbb{R}$ with $M+1$ uniform knots $\nu_m = (m - M/2)\Delta, m = 0, \cdots, M$, where Δ corresponds to the spacing between the knots. Then, the learnable parameters of σ_i are $c^i = (c^i_m)_{m=1}^M$ s.t. $\sigma_i(\nu_m) = c^i_m$. We denote now σ_i by σ_{c^i} to underline the links between c_i and σ_i. And, to fully characterize σ_{c^i}, the extensions are defined as follows:

$$\begin{aligned}\sigma_{c^i}(x) &= c^i_0, x \in (-\infty, \nu_0], \\ \sigma_{c^i}(x) &= c^i_M, x \in [\nu_M, \infty).\end{aligned}$$

To ensure the first constraint (1), we need $c^i_m - c^i_{m-1} \geq 0$ for $m = 1, \cdots M$ and for all i. So, instead of considering σ_{c^i}, we consider $\sigma_{P_\uparrow(c^i)}$, where $P_\uparrow$ is the projection onto the feasible set $\{c \in \mathbb{R}^{M+1} : c_m - c_{m-1} \geq 0, \forall m = 1, \cdots M\}$. In fact, $P_\uparrow$ just preserves the non-negative differences and set the negative ones to 0.

To avoid overfitting on the trainable set and losing generalizability to arbitrary linear operators H, a sparsity promoter on the coefficients of the linear splines is added. It penalizes the second order total variation of each linear spline $\sigma_{P_\uparrow(c^i)}$. The relative regularizer for each linear spline is $\|LP_\uparrow(c^i)\|_1$, where $L \in \mathbb{R}^{(M-1)\times(M+1)}$ is the

second-order finite-difference matrix. More details are given in [4]. Hence, we obtain a new final loss, which is:

$$\sum_{i=1}^{I} \mathcal{L}(x_i, T^t_{R^C_\theta,\lambda,\alpha}(y_i)) + \eta \sum_{j=1}^{c} \|LP_\uparrow(c_i)\|_1, \tag{21}$$

where η is a hyperparameter which tunes the strength of the regularization. We focus now on the step size α which appears in (18). To ensure that the iterations of the gradient descent are converging, we should restrict $\alpha \in (0, \frac{2}{1+\lambda L_\theta})$ as mentioned above. In fact, this restriction guarantees that the t-fold composition of $T^t_{\mathbb{R}^C_\theta,\lambda,\alpha}$ computes the actual minimizer of (14). The authors proposed to learn a sharp bound of L_θ, the Lipschitz constant of $\nabla R^C_\theta(\cdot)$ to enforce the stability of the training, even for small t (more details are given in *Proposition 4.1*, [10]).
The authors also proposed to add one last tunable scaling parameter $\mu \in \mathbb{R}^+$ with the aim to boost the universality (w.r.t the standard deviation of the noise) of the CRRNN. Then, the inverse problem becomes:

$$x^* = \underset{x\in\mathbb{R}^{n\times m}}{\arg\min} \|H(x) - y\|_2^2 + \frac{\lambda}{\mu} R^C_\theta(\mu x). \tag{22}$$

Then, the t-step operator $T^t_{R^C_\theta,\alpha,\lambda}(\cdot)$ becomes:

$$T_{R^C_\theta,\alpha,\lambda,\mu} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m} \tag{23}$$

$$x \mapsto x - \alpha((x - y) + \lambda \nabla R^C_\theta(\mu x)), \tag{24}$$

where now $\alpha < \frac{2}{1+\lambda\mu L_\theta}$.

2.2.2 Theoretical guarantees

By restricting the profiles ψ_i to be convex and having a Lipschitz continuous derivative, it has been theoretically proven that:

$$\emptyset \neq \underset{x\in\mathbb{R}^{n\times m}}{\arg\min} \frac{1}{2}\|H(x) - y\|_2^2 + \sum_{i=1}^{c} \psi_i(w_i^T \mathbf{x}), \tag{25}$$

for any arbitrary linear operator $H(\cdot)$ and given that $\arg\min_{t\in\mathbb{R}} \psi_i(t) \neq \emptyset, \forall i = 1, \cdots, c$ *(Proposition 3.1, [10])*. So, the existence of a minimizer is established. It leads then to the analysis about the stability of the reconstruction which has also been theoretically proven in (*Proposition 3.2*, [10]), i.e. for any $y_1, y_2 \in \mathbb{R}^{p\times q}$, the following statement holds:

$$\|H(x_1) - H(x_2)\|_2^2 \leq \|y_1 - y_2\|_2^2, \tag{26}$$

where $x_j \in \underset{x\in\mathbb{R}^{n\times m}}{\operatorname{argmin}} \frac{1}{2}\|H(x) - y_j\|_2^2 + \sum_{i=1}^{c} \psi_i(w_i^T \mathbf{x})$ and $j = 1, 2$.
The regularizer $R^C_\theta(\cdot)$ is not explicitly needed in the reconstruction scheme unlike its gradient ∇R^C_θ. Hence, it makes more sense to work directly on the gradient in order to choose a convenient structure on ψ_i. Therefore, a structure is chosen explicitly on $\sigma_i := \psi_i'$. The authors proposed to pick σ_i in the subset $\mathcal{LS}^t_\uparrow \subset C^{0,1}_\uparrow$ which corresponds to the subset of increasing linear splines with at most t knots. Hence, the following statement has been proved (*Proposition 3.5*, [10]) which underlines the expressivity of a such choice of activation functions for σ_i:
For any compact subset $\Omega \subset \mathbb{R}^{nm}$ and $t \geq 2$, the set

$$\left\{\mathbf{W}^T \sigma(\mathbf{W}\cdot) : \mathbf{W} \in \mathbb{R}^{c\times nm}, \sigma_i \in \mathcal{LS}^t_\uparrow(\mathbb{R})\right\} \tag{27}$$

is dense in

$$\left\{\mathbf{W}^T \sigma(\mathbf{W}\cdot)\big|_\Omega : \mathbf{W} \in \mathbb{R}^{c\times nm}, \sigma_i \in C^{0,1}_\uparrow\right\} \tag{28}$$

with respect to the norm $\|f\|_{C(\Omega)} := \sup_{x\in\Omega}\|f(x)\|$.
This result allows us to claim the following statement:
For any compact and convex subset $\Omega \subset \mathbb{R}^{nm}$, the regularizers of the form (13) s.t. their gradient belongs to the set (27), are dense in the set

$$\left\{\sum_{i=1}^{c} \psi_u(w_i^T \mathbf{x}) : \psi_i \in C^{1,1}(\mathbb{R}) \text{ convex}, w_i \in \mathbb{R}^{nm}\right\} \tag{29}$$

with respect to the norm $\|f\|_{C^1(\Omega)} \coloneqq \sup_{x\in\Omega}\|f(x)\| + \sup_{x\in\Omega}\|J_f(x)\|$. Note that the norm $\|\cdot\|_{C^1(\Omega)}$ is stronger than the norm $\|\cdot\|_{C(\Omega)}$.
Hence, all of these theoretical statements lead the authors to parameterize the learnable profiles function σ_i with learnable linear-spline activation functions. It implies then that the convex profiles ψ_i are splines of degree 2. So, this theoretical analysis shows that even if a neural network architecture is used, we can still have theoretical guarantees concerning the convergence and the stability of the reconstruction related to the regularizer $R^C_\theta(\cdot)$. They showed though that parametrezing the profiles σ_i with the popular activation function in the deep learning framework **ReLu**$(\cdot)$ leads to a loss in expressivity. Actually, the set

$$\left\{\mathbf{W}^T\mathbf{ReLU}(\mathbf{W}\cdot - b) : \mathbf{W}\in\mathbb{R}^{c\times nm}, b\in\mathbb{R}\right\} \tag{30}$$

is not dense in (28). However, we note that every function of the shape **ReLu** can be approximated by a linear spline on a defined grid.

2.2.3 Reconstruction method

Suppose that we have a trained CRRNN model $R^C_{\theta^*_1}$ and the goal is to reconstruct $x^* = \arg\min_{x\in\mathbb{R}^{n\times m}}\|H(x)-y\|_2^2 + \frac{\lambda}{\mu}R^C_{\theta^*_1}(\mu x)$, where λ, μ are the tunable parameters and are not necessarily equal to their learnt value λ^*_1, μ^*_1. In fact, those parameters have been introduced in the learning process with the unique aim to boost the performance. Many algorithm schemes allow us to retrieve x^* such as *FISTA*. However, we choosed to use the Adaptive accelerated gradient descent algorithm *AdaGD* which is decribed as follows [13]:

Algorithm 2 AdaGD

Require: $x^0\in\mathbb{R}^{n\times m}, \lambda_0 > 0, \Lambda_0 > 0, s_0 = S_0 = +\infty$
  $y^1 = x^1 = x^0 - \lambda_0\nabla f(x^0)$
  for $k = 1, 2, \cdots$ **do**
    $\lambda_k = \min\left\{\sqrt{1+\frac{s_{k-1}}{2}}\lambda_{k-1}, \frac{\|x^k - x^{k-1}\|_2}{2\|\nabla f(x^k)-\nabla f(x^{k-1})\|_2}\right\}$
    $\Lambda_k = \min\left\{\sqrt{1+\frac{S_{k-1}}{2}}\Lambda_{k-1}, \frac{\|\nabla f(x^k)-\nabla f(x^{k-1})\|_2}{2\|x^k - x^{k-1}\|_2}\right\}$
    $\beta_k = \frac{\sqrt{1/\lambda_k}-\sqrt{\Lambda_k}}{\sqrt{1/\lambda_k}+\sqrt{\Lambda_k}}$
    $y^{k+1} = x^k - \lambda_k\nabla f(x^k)$
    $x^{k+1} = y^{k+1} + \beta_k(y^{k+1} - yk)$
    $s_k = \frac{\lambda_k}{\lambda_{k-1}}, S_k = \frac{\Lambda_k}{\Lambda_{k-1}}$
  end for

, where $f(\cdot) = \|H(\cdot) - y\|_2^2 + \lambda/\mu R^C_{\theta^*_1}(\cdot)$ and, then, $\nabla f(\cdot) = H^t(H(x) - y) + \lambda\nabla R^C_{\theta^*_1}(\mu\cdot)$.

2.3 Reconstruction method of the infimal convolution regularizer

Back to the main convex infimal convolution regularizer, the inverse problem to solve is the following:

$$x^* = \arg\min_{x\in\mathbb{R}^{n\times m}}\min_{g\in\mathbb{R}^{n\times m}}\frac{1}{2}\|x-y\|_2^2 + R^{\text{infConv}}_{\alpha,\beta,\theta}(x) + \chi_+(g),\ \text{s.t. } x = g, \tag{31}$$

where the function $\chi_+(\cdot)$ enforces the positivity of the output since we are working with discrete representation of images as done in [15]. We observe that (31) can be rewritten as follows:

$$\arg\min_{x\in\mathbb{R}^{n\times m}}\min_{z,w,g\in\mathbb{R}^{n\times m}}\frac{1}{2}\|x-y\|_2^2 + R_{TV}(z) + \beta R^C_\theta(w) + \chi_+(g), \tag{32}$$

$$\text{s.t. } -u + z + w = 0 \text{ and } u - g = 0 \tag{33}$$

Hence, the constraints in (33) can be formulated under a matrix-vector equation as follows:

$$\begin{bmatrix} -I & I & I & 0\\ I & 0 & 0 & -I\end{bmatrix}\begin{bmatrix} u\\ z\\ w\\ g\end{bmatrix} = \mathbf{0},$$

which is equivalent to the following equation:

$$A_u u + A_z z + A_w w + A_g g = 0 \in \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m},$$

where $A_u = [-I, I]^T, A_z = [I, 0]^T, A_w = [0, I]^T$ and $A_g = [0, -I]^T$. We can reconstruct x^* using an ADMM scheme.

2.3.1 Steps of the ADMM scheme

Given some initial value to $u_0, w_0, z_0, g_0 \in \mathbb{R}^{n\times m}$ and $\Theta_0 = [\Theta_1, \Theta_2]^T \in \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m}$, which correspond to the scaled dual variables, the iterations are described as follows for $k \geq 0$:

$$u^{k+1} = \underset{u\in\mathbb{R}^{n\times m}}{\arg\min} \frac{1}{2}\|u - y\|_2^2 + \frac{\lambda}{2}\|A_u u + A_z z^k + A_w w^k + A_g g^k + \Theta^k\|_2^2, \tag{34}$$

$$z^{k+1} = \underset{z\in\mathbb{R}^{n\times m}}{\arg\min} \alpha R_{TV}(z) + \frac{\lambda}{2}\|A_u u^{k+1} + A_z z + A_w w^k + A_g g^k + \Theta^k\|_2^2, \tag{35}$$

$$w^{k+1} = \underset{w\in\mathbb{R}^{n\times m}}{\arg\min} \beta R_\theta^C(w) + \frac{\lambda}{2}\|A_u u^{k+1} + A_z z^{k+1} + A_w w + A_g g^k + \Theta^k\|_2^2, \tag{36}$$

$$g^{k+1} = \underset{g\in\mathbb{R}^{n\times m}}{\arg\min} \chi_+(g) + \frac{\lambda}{2}\|A_u u^{k+1} + A_z z^{k+1} + A_w w^{k+1} + A_g g + \Theta^k\|_2^2, \tag{37}$$

$$\Theta^{k+1} = \Theta^k + A_u u^{k+1} + A_z z^{k+1} + A_w w^{k+1} + A_g g^{k+1}, \tag{38}$$

where $\lambda > 0$ is the Lagrangian multiplier. Hence, we end up with four individuals optimization problems at each iteration. Hopefully, there exists fast methods to solve each of them.

u-update: We first observe that:

$$\begin{aligned}
(34) &\iff u^{k+1} = \underset{u\in\mathbb{R}^{n\times m}}{\arg\min} \frac{1}{2}\|u - y\|_2^2 + \frac{\lambda}{2}\|-u + z^k + w^k + \Theta_1^k\|_2^2 + \frac{\lambda}{2}\|u - g^k + \Theta_2^k\|_2^2, \\
&\iff u^{k+1} = \underset{u\in\mathbb{R}^{n\times m}}{\arg\min} \frac{1}{2}\left\|\begin{bmatrix} I \\ \sqrt{\lambda}I \\ \sqrt{\lambda}I \end{bmatrix} u - \begin{bmatrix} y \\ \sqrt{\lambda}(z^k + w^k + \Theta_1^k) \\ \sqrt{\lambda}(g^k - \Theta_2^k) \end{bmatrix}\right\|_2^2 \\
&\iff u^{k+1} = (I + 2\lambda I)(y + \lambda(z^k + w^k + \Theta_1^k + g^k - \Theta_2^k)) \\
&\iff u^{k+1} = \frac{1}{2\lambda + 1}(y + \lambda(z^k + w^k + \Theta_1^k + g^k - \Theta_2^k)),
\end{aligned}$$

where the third line comes from the derivation of the normal equation. Hence, the computation is direct here.

z-update: We get

$$\begin{aligned}
(35) &\iff z^{k+1} = \underset{z\in\mathbb{R}^{n\times m}}{\arg\min} \alpha R_{TV}(z) + \frac{\lambda}{2}\left\|z - u^{k+1} + w^k + \Theta_1^k\right\|_2^2 \\
&\iff z^{k+1} = \underset{z\in\mathbb{R}^{n\times m}}{\arg\min} \frac{\alpha}{\lambda} 2R_{TV}(z) + \left\|z - u^{k+1} + w^k + \Theta_1^k\right\|_2^2.
\end{aligned}$$

Hence this can be solved by sufficiently fast using *FISTA* algorithm for TV image denoising explained in [2] or in section 2.1.1. We approximate the proximal map of Moreau with the *FGP* algorithm. In fact, the proximal map of Moreau suffices as explained in [8]. We should pay attention that this minimization has to be on $\mathbb{R}^{n\times m}$, not on a constrained subset $\mathcal{E} \subset \mathbb{R}^{n\times m}$ like it is usually done..

w-update: We get

$$\begin{aligned}
(36) &\iff w^{k+1} = \underset{w\in\mathbb{R}^{n\times m}}{\arg\min} \beta R_\theta^C(w) + \frac{\lambda}{2}\left\|w - (u^{k+1} - z^{k+1} - \Theta_1^k)\right\|_2^2 \\
&\iff w^{k+1} = \underset{w\in\mathbb{R}^{n\times m}}{\arg\min} \frac{\beta}{\lambda} R_\theta^C(w) + \frac{1}{2}\left\|w - (u^{k+1} - z^{k+1} - \Theta_1^k)\right\|_2^2
\end{aligned}$$

To solve this problem, we compute easily the gradient step operator for a step size α, which is equal to

$$\begin{aligned}&(1-\alpha)w + \alpha((u^{k+1} - z^{k+1} - \Theta_1^k) - \frac{\beta}{\lambda}\nabla R_\theta^C(w))\\ =&(1-\alpha)w + \alpha((u^{k+1} - z^{k+1} - \Theta_1^k) - \frac{\beta}{\lambda}\mathbf{W}^T\sigma(\mathbf{W}w)).\end{aligned}$$

Then, we apply the *AdaGD* algorithm as described in section 2.2.3.

g-update: We get

$$\begin{aligned}(37) \iff g^{k+1} &= \underset{g\in\mathbb{R}^{n\times m}}{\arg\min}\, \chi_+(g) + \frac{\lambda}{2}\left\|u^{k+1} - g + \Theta_2^k\right\|_2^2\\ \iff g^{k+1} &= max(0, u^{k+1} + \Theta_2^k)\end{aligned}$$

The computation is explicit here.

More details on the ADMM algorithm are described in [5]. In practice, those iterations converges quickly to a convenient accuracy but takes a lot of time to reach a high accuracy.

3 Implicit Layers

In the t-unfolding gradient descent framework, we do not fully solve the inverse problem. Thus, we apply exactly the same process to every batch of the training data set. However, if we fully solve every batch in a dummy way, the number of iterations would vary from a batch to another. Thus, the process applied would not be the same across the batches. And, remark that the number of iterations would be very large which is a major issue, because we rely on the basic automatic differentiation, a lot of intermediate iterates would have to be saved to be able to compute the final gradient. So, we would encounter serious issues related to the performance of the computations and the backpropagation could be very unstable numerically (see [12] for more details). To overcome these issues, we can refer to the implicit layers method.

3.1 Implementation of implicit layers

We are used to layers with an explicit forward pass, i.e.

$$z = f(x),$$

, where x is the input of the layer and z is the output. However, one could also think of an implicit layer where we are looking for the output z to satisfy some constraint e.g., finding the root of an equation z s.t.

$$g(x, z) = 0.$$

Hence, this formulation allows us to separate the solution procedure of the layer from the definition of the layer itself. This establishes the key step for the differentiation. To compute the derivative $\frac{\delta z}{\delta x}$, we can directly apply the implicit function theorem. Thus, the derivative is completely independent of the method applied to compute z. Let's get through an simple example to be clearer.

Suppose that the layer is specified s.t. we are looking for $z(x) \in \mathbb{R}$ which satisfies

$$\begin{aligned} z(x) &= sin(wz(x) + x) \\ 0 &= sin(wz(x) + x) - z(x), \end{aligned}$$

for some value of the input $x \in \mathbb{R}$. To compute $z(x)$, we could simply apply an iteration algorithm to some initial value z_0 and compute $z_t = sin(wz_{t-1})$ until convergence. If this was under the explicit layer framework, to compute the derivative $\frac{\delta z_T}{\delta x}$ via automatic differentiation, we should keep in memory all values $\frac{\delta z_t}{\delta z_{t-1}}$ to finally compute through backpropagation

$$\frac{\delta z_T}{\delta x} = \frac{\delta z_T}{\delta z_{T-1}} \cdot \frac{\delta z_{T-1}}{\delta z_{T-2}} \cdots \frac{\delta z_1}{\delta x}.$$

Hence, if we are working in a high dimensional space (which is often the case in deep learning and in computer vision), the Jacobian matrices $\frac{\delta z_t}{\delta z_{t-1}}$ will be very large matrices that we will need to store until the backpropagation which happened only after a sufficient number of iterations. Hence, the memory consumption will increase as long as it has not converged. And, it get even worse when we are working with batches of data. To counteract this issue, we apply the implicit function theorem which states:

Proposition 3.1. *Let $f : \mathbb{R}^p \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ and $x_0 \in \mathbb{R}^p, z_0 \in \mathbb{R}^n$ be such that*

- $f(x_0, z_0) = 0$,
- *f is continuously differentiable with non-singular Jacobian $\frac{\delta f(x_0, z_0)}{\delta z_0} \in \mathbb{R}^{n \times n}$.*

Then, there exist open sets $\mathcal{S}_{x_0} \subset \mathbb{R}^p$ ans $\mathcal{S}_{z_0} \subset \mathbb{R}^n$ containing x_0 and z_0, respectively, and a unique continuous function $z^* : \mathcal{S}_{x_0} \longrightarrow \mathcal{S}_{z_0}$ such that

- $z_0 = z^*(x_0)$,
- $f(x, z^*(x)) = 0, \quad \forall a \in \mathcal{S}_{x_0}$, *and*
- *z^* is differentiable on $\mathcal{S}_{x_0}$.*

Hence, the Jacobian is computed as follows.

$$\frac{\delta z^*(x)}{\delta x} = -\left[\frac{\delta f(x, z_x)}{\delta z_x}\right]^{-1} \frac{\delta f(x, z_x)}{\delta x}. \tag{39}$$

So, we compute a solution $z^* = z^*(x)$ with the most convenient algorithm such that $z^* - sin(wz^* + x) = 0$ out of the automatic differentiation tape, i.e. under the `torch.no_grad()` context manager in PyTorch and compute the Jacobian described in (39) to insert it in the computational graph of automatic differentiation using the registration of hooks.

Under the CRRNN framework, a denoised image x^* is a solution of:

$$x^* \in \underset{x \in \mathbb{R}^{n \times m}}{\arg\min} \frac{1}{2}\|x - y\|_2^2 + \frac{\lambda}{\mu} R_\theta^C(\mu x), \tag{40}$$

where λ and μ are the tunable hyperparameters. Hence, x^* satisfy the following because of the differentiability of the objective function to minimize:

$$(x^* - y) + \lambda \nabla R_\theta^C(\mu x^*) = 0. \tag{41}$$

Hence, for each batch in the training set, we full solve the images through the AdaAGD algorithm out of the scope of the automatic differentiation. Then, for each image x, we evaluate $x = x - ((x - y) + \lambda \nabla R_\theta^C(\mu x))$ in the scope now and compute the Jacobian $J_x = W^T(W \otimes \varphi''(Wx))$ explicitly and move to the "register hook" step.

3.2 Numerical Results

Two CRRNN models have been trained under the same procedure as in [10], i.e. two training noises have been used $\sigma \in \{\frac{5}{255}, \frac{25}{255}\}$, 238'400 patches of size (40×40) from the 400 images of the BSD500 dataset have been brought together to form 128 batches. More details concerning the training procedure can be found in (section VI, [10]).

We first compare the implicit layers and the t-unfolding gradient descent with $t \in \{5, 10\}$ by evaluating the denoising task of different CRRNN models on the BSD68 test set of grayscale images. (The pretrained CRRNN models through t-unfolding gradient descent are from [10].) To do so, a noise with standard deviation of $\sigma = \frac{10}{255}$ has been added. For each regularizer tested, the hyperparameter λ used is the one which maximises the performances through coarse-to-fine method and μ is set to 1 for the CRRNN models. The adaptive accelerated gradient descent algorithm *AdaGD* (from [13]) has been used to solve the inverse problem related to the CRRNN models.[1][2]

Performances of regularizers		
Regularizer	psnr	ssim
Total variation	29.86	0.81
CRRNN, $t = 5, \sigma = \frac{5}{255}$	32.25	0.88
CRRNN, $t = 10, \sigma = \frac{5}{255}$	32.25	**0.89**
CRRNN, $t = 5, \sigma = \frac{25}{255}$	25.31	0.88
CRRNN, $t = 10, \sigma = \frac{25}{255}$	29.35	0.81
CRRNN, Implicit Layers, $\sigma = \frac{5}{255}$	**32.33**	0.88
CRRNN, Implicit Layers, $\sigma = \frac{25}{255}$	28.11	0.78

Table 1: Under the denoising task ($\sigma = \frac{10}{255}$), the CRRNN models are better than the TV regularizer. Among the CRRNN models, the training noise $\sigma = \frac{5}{255}$ should be preferred to $\sigma = \frac{25}{255}$. Implicit layers approach is slightly better than the t-unfolding gradient method under the training noise $\sigma = \frac{5}{255}$.

One observation one can draw from figures (4) and (5) is the scale of the values of the different regularization costs of the two CRRNN models. We remark that they do not seem to be on the same scale when we look at the values of the regularization costs of the clean and noisy images.

[1] All the code of the Python implementation and experiments can be found in `https://github.com/Waegaf/PDM`.

[2] For every numerical denoising task, the hyperparameter selected manually is the one which maximized the performance.

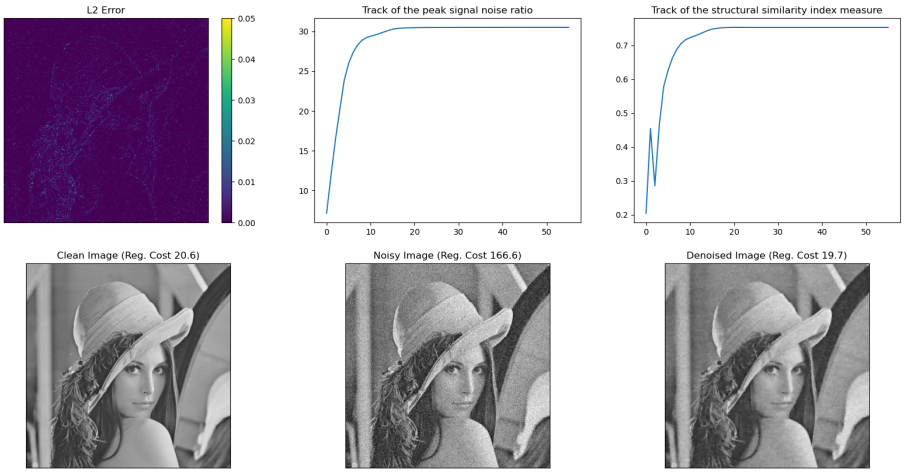

Figure 4: *Bottom*: *Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image through the CRRNN model trained via implicit layers and under training noise of $\sigma = \frac{5}{255}$. *Top*:*Left*: The L2 error between the clean and denoised image. *Middle* and *Left*: The track of the psnr and ssim metrics along the iterations of the accelerated gradient descent algorithm.

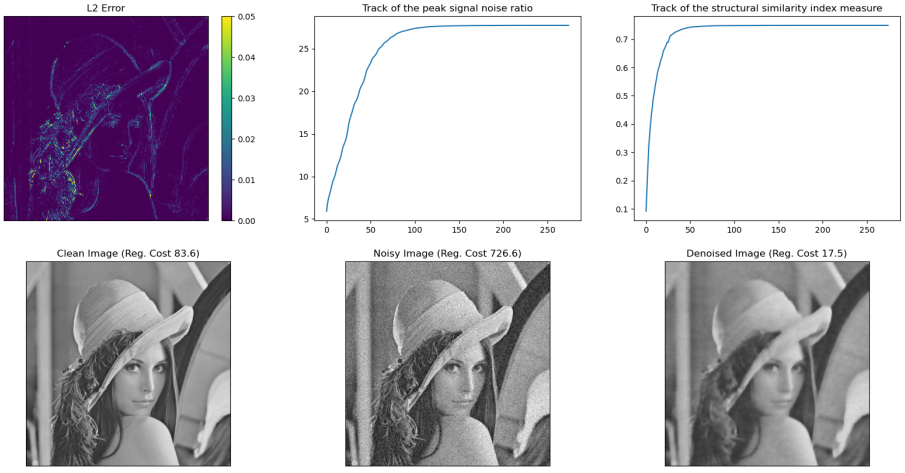

Figure 5: *Bottom*: *Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image through the CRRNN model trained via implicit layers and under training noise of $\sigma = \frac{25}{255}$. *Top*:*Left*: The L2 error between the clean and denoised image. *Middle* and *Left*: The track of the psnr and ssim metrics along the iterations of the accelerated gradient descent algorithm.

Now, we compare the performances of the same CRRNN models used above under the infimal convolution framework using exactly the same procedure as above. Instead of the *AdaGD*, we use the ADMM algorithm. We have set $\alpha = 5 \cdot 10^{-2}$ and β corresponds to the best hyperparameter which achieved the result above (for each CRRNN model) to construct the $R^{infConv}_{\alpha,\beta,\theta}$ regularizer.

Performances of regularizers under the inf. conv. framework		
Regularizer	psnr	ssim
CRRNN, $t = 5, \sigma = \frac{5}{255}$	29.63	0.77
CRRNN, $t = 10, \sigma = \frac{5}{255}$	29.62	0.77
CRRNN, $t = 5, \sigma = \frac{25}{255}$	29.56	0.77
CRRNN, $t = 10, \sigma = \frac{25}{255}$	29.58	0.77
CRRNN, Implicit Layers, $\sigma = \frac{5}{255}$	**29.76**	0.77
CRRNN, Implicit Layers, $\sigma = \frac{25}{255}$	29.56	0.77

Table 2: Under the infimal convolution framework, the performances are worst than the CRRNN regularizers and the TV regularizer. Surprisingly, the ssim value is stucked at 0.77. However, the CRRNN model trained via implicit layers with $\sigma = \frac{5}{255}$ has the best performances among the other CRRNN models.

Concerning the ssim score of table (2), we observed that varying the hyperparameter related to the TV regularizer make also vary the ssim score for each regularizer. We present now some visual results of the denoising task under the convex infimal convolution framework using different CRRNN models.

Figure 6: *Top*: *Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image through a CRRNN model trained via 5-unfolding gradient descent and under a training noise of $\frac{5}{255}$. *Bottom*: *Left*: The denoised image through a CRRNN model trained via 5-unfolding gradient descent and under a training noise of $\frac{25}{255}$. *Middle*: The denoised image through a CRRNN model trained via implicit layers and under a training noise of $\frac{5}{255}$.*Left*: The denoised image through a CRRNN model trained via implicit layers and under a training noise of $\frac{25}{255}$.

Figure 7: *Top*: *Left*: The clean image with strides. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image through a CRRNN model trained via 5-unfolding gradient descent and under a training noise of $\frac{5}{255}$. *Bottom*: *Left*: The denoised image through a CRRNN model trained via 5-unfolding gradient descent and under a training noise of $\frac{25}{255}$. *Middle*: The denoised image through a CRRNN model trained via implicit layers and under a training noise of $\frac{5}{255}$.*Left*: The denoised image through a CRRNN model trained via implicit layers and under a training noise of $\frac{25}{255}$.

4 Training of a CRRNN model under the infimal convolution framework

Here, we train here a CRRNN model under the infimal convolution framework with the TV regularizer. To do so, we used a mixed of two known methods: t-unfolding gradient descent and implicit layers. In fact, instead of using a gradient scheme here, we are solving through ADMM algorithm. Hence, we apply the t-unfolding of the ADMM algorithm. More formally, we define the following operator:

$$T_{\theta,\lambda,\alpha,\beta}(u, z, w, g, \Theta_1, \Theta_2) = (u', z', w', g', \Theta_1', \Theta_2'), \tag{42}$$

where

$$\begin{aligned}
u' &= \underset{u\in\mathbb{R}^{n\times m}}{\arg\min} \frac{1}{2}\|u - y\|_2^2 + \frac{\lambda}{2}\|-u + z + w + \Theta_1\|_2^2 + \frac{\lambda}{2}\|u - g + \Theta_2\|_2^2, && (43)\\
&= \frac{1}{2\lambda + 1}(y + \lambda(z + w + \Theta_1 + g - \Theta_2) && (44)\\
z' &= \underset{z\in\mathbb{R}^{n\times m}}{\arg\min}\, \alpha R_{TV}(z) + \frac{\lambda}{2}\|-u' + z + w + \Theta_1\|_2^2, && (45)\\
w' &= \underset{w\in\mathbb{R}^{n\times m}}{\arg\min}\, \beta R_\theta(w) + \frac{\lambda}{2}\|-u' + z' + w + \Theta_1\|_2^2, && (46)\\
g' &= \underset{g\in\mathbb{R}^{n\times m}}{\arg\min}\, \chi_+(g) + \frac{\lambda}{2}\|u' - g + \Theta_2\|_2^2, && (47)\\
&= max(0, u' + \Theta_2) && (48)\\
\Theta_1' &= \Theta_1 + (-u' + z' + w'), && (49)\\
\Theta_2' &= \Theta_2 + (u' - g'). && (50)
\end{aligned}$$

Hence, the t-step operator $T^t_{\theta,\lambda,\alpha,\beta} = T_{\theta,\lambda,\alpha,\beta} \circ \cdots \circ T_{\theta,\lambda,\alpha,\beta}$ corresponds to a t composition of (42), where θ corresponds to the parameters of the CRRNN R_θ, α is the hyperparameter coefficient of the TV, β is the hyperparameter coefficient of the CRRNN regularizer and λ corresponds to the Lagrange multiplier. Ideally, we should learn θ such that the output u of $T^\infty_{\theta,\lambda,\alpha,\beta}$ is the denoised image. However, by the same arguments used in [10], we can approximate $T^\infty_{\theta,\lambda,\alpha,\beta}$ with a finite number of step t. The implicit layers method is used within the application of the operator $T_{\theta,\lambda,\alpha,\beta}$, when we have to solve (45) and (46), the remaining ones can be explicitly computed.

4.1 Numerical Result

We trained two CRRNN models with different training noises $\sigma \in \{\frac{5}{255}, \frac{25}{255}\}$ through the 1-unfolding ADMM algorithm. Those models have been trained under the same procedure described above for at least 3 epochs. The choice to approximate $T^\infty_{\theta,\lambda,\alpha,\beta}$ by $T^1_{\theta,\lambda,\alpha,\beta}$ is due to time reasons. In fact, around a week of training is needed to train 3 epochs on a Nvidia gpu.

Performances of regularizers under the convex inf. conv. framework		
Regularizer	psnr	ssim
CRRNN, $\sigma = \frac{5}{255}$	23.56	0.77
CRRNN, $\sigma = \frac{25}{255}$	23.56	0.77

Table 3: Here are the results of the performances of the two convex infimal convolution regularizers with the both trained CRRNN models under the $T^1_{\theta,\lambda,\alpha,\beta}$ framework. To compare it to table (2), the ssim score is again the same and equal to 0.77. The psnr score is the same for the both models and several convex infimal convolution with a pretrained CRRNN model have a higher score.

We present here some visual results of the denoising task under the convex infimal convolution regularizer using the both trained CRRNN models.

Figure 8: *Top*:*Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image via the convex infimal convolution regularizer with a trained CRRNN model trained under the training noise $\sigma = \frac{5}{255}$. *Bottom*:*Left*: The clean image with strides. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image via the convex infimal convolution regularizer with a trained CRRNN model trained under the training noise $\sigma = \frac{5}{255}$.

Figure 9: *Top*:*Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image via the convex infimal convolution regularizer with a trained CRRNN model trained under the training noise $\sigma = \frac{25}{255}$. *Bottom*:*Left*: The clean image with strides. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$. *Right*: The denoised image via the convex infimal convolution regularizer with a trained CRRNN model trained under the training noise $\sigma = \frac{25}{255}$.

5 Infimal Convolution Regularizer: Weakly Convex case

Now, we focus on the weakly convexity case. First of all, we define the weak and strong convexity in the regularizers framework.

Definition 5.1. *Let $\rho > 0$. A function $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ is said to be ρ-weakly convex if the function $x \mapsto \frac{\rho}{2}\|x\|_2^2 + f(x)$ is a convex function.*

Definition 5.2. *Let $\rho > 0$. A function $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ is said to be ρ-strongly convex if the function $x \mapsto -\frac{\rho}{2}\|x\|_2^2 + f(x)$ is a convex function.*

To have a better understanding and intuition on weakly convex functions, we state some equivalent definitions in the following proposition from [7].

Proposition 5.1. *Given that $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ and $\rho \geq 0$, the following properties are equivalent:*

(a) f is ρ-weakly convex,

(b) f satisfies

$$\lambda f(x) + (1-\lambda)f(y) - f(\lambda x + (1-\lambda)y) \geq -\rho\frac{\lambda(1-\lambda)}{2}\|x-y\|_2^2, \tag{51}$$

for all $x, y \in \mathbb{R}^{n\times m}$ and for all $\lambda \in [0,1]$.

(c) f satisfies

$$f(x+h) + f(x-h) - 2f(x) \geq -C|h|^2, \tag{52}$$

for all $x, h \in \mathbb{R}^{n\times m}$.

(d) There exists two functions $f_1, f_2 : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ such that $f = f_1 + f_2$, where f_1 is convex and $f_2 \in C^2(\mathbb{R}^{n\times m})$ and satisfies $\|D^2 f\|_\infty \leq \rho$.

Hence, intuitively, a ρ-weakly convex function behaves as it is convex but can dispose of some concave "bumps" whose size is controlled by ρ. Now, we show an important result concerning the property of infimal convolution that will be useful in the weak convexity framework. In fact, this will be the basis of the weak convex infimal convolution regularizer.

Proposition 5.2. *Let $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ be $\frac{\mu}{2}$-weakly convex and $g : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ be μ-strongly convex with $\mu > 0$. If the infimal convolution $f\square g$ is exact, i.e. for all $x \in \mathbb{R}^{n\times m}$, there exists $y_x \in \mathbb{R}^{n\times m}$ such that:*

$$f\square g(x) = \inf_{y\in\mathbb{R}^{n\times m}}\big(f(x-y) + g(y)\big) \tag{53}$$

$$= f(x-y_x) + g(y_x). \tag{54}$$

Then, $f\square g$ is μ-weakly convex.

Proof. See proof in Appendix A. □

Different ways to construct a weakly convex infimal convolution regularizer exist. For example, we can show that the infimal convolution between a weakly convex function and a convex function is weakly convex if the following inequation holds for every $x, x' \in \mathbb{R}^{n\times m}$;

$$\|y_x - y_{x'}\|_2^2 \leq C\|x - x'\|_2^2,$$

where C is a non-negative constant.
Hence, we can construct the following infimal convolution regularizer by considering $f(x) = \alpha R_{TV}^{\mu}(x)$ a strongly convex approximation of the TV regularizer and $g(x) = \beta R_\theta^W(x)$ and work with the following regularizer:

$$R_\theta^{WInfConv}(x) = \min_{y\in\mathbb{R}^{n\times m}}(R_{TV}^{\mu}(x-y) + R_\theta^W(y)). \tag{55}$$

5.1 Strongly convex approximation of the TV

Many variants of the standard TV regularizer have emerged in the last decades [6]. We focus here on the strongly convex approximation R^{μ}_{TV} which operates as follows:

$$R^{\mu}_{TV}(x) = \sqrt{\|\nabla x\|_2^2 + \mu} \tag{56}$$

, where $x \in \mathbb{R}^{n\times m}$ corresponds to the discrete representation of an image. This approximation of the TV regularizer is sometimes used to avoid the non-differentiability issues of the TV regularizer. Hence, μ should be very small to have a convenient approximation. We can show that R^{μ}_{TV} is a C_μ-strongly convex function with $C_\mu > 0$ by considering that we are working with discrete representation of images space (see more details in Appendix B).

Unfortunately, it is very difficult to estimate C_μ or a tight lower bound. In fact, there are not so many strongly convex regularizers since these are more restricted regularizers than convex ones with no specific advantages. Moreover, μ should be very small to keep the approximation to the TV regularizer good enough. Please note that according to some experts [9], this approach should not be used if there is no need to have a strongly convex regularizer.

5.1.1 Reconstruction method

We use here the major advantage of this TV approximation R^{μ}_{TV}: its differentiability. In fact, we avoid any division by 0 by adding this small non-negative constant μ. Hence, any gradient based algorithm are convenient to solve the relative inverse problem. So, the *AdaGd* algorithm is adequate here. We could also use the *FISTA* algorithm since the gradient of $x \mapsto \frac{1}{2}\|x-y\|_2^2 + R^{\mu}_{TV}(x)$ is $\frac{1}{\mu}$-Lipschitz (Section 3.3,[9]).

5.2 Weakly convex ridge regularizer neural network

The WCRRNN model R^{W}_{θ} from [11] is very close to the CRRNN model. The regularity of an image is measured as follows:

$$R^{W}_{\theta} : x \mapsto \sum_{i=1}^{N_C} \sum_{k\in\mathbb{Z}^2} \psi_i((h_i * x)[k]). \tag{57}$$

As in the CRRNN framework, we can reformulate in a generic form,

$$R^{W}_{\theta} : x \mapsto \sum_{i} \psi_i(w_i^T \mathbf{x}), \tag{58}$$

where ψ_i are 1-weakly convex functions and the spectral norm of $W = [w_1, \cdots, w_p]$ is equal to 1.
The major differences with the CRRNN framework is the parametrization of the activation functions ψ_i and the restriction of the convolutional neural network related s.t. its matrix W satisfies the above property. So, each activation function ψ_i is parametrized as follows:

$$\psi_{(\cdot)} = \frac{\psi(\alpha_i \cdot)}{\alpha_i^2},$$

where ψ is a learnable linear spline and $\frac{e^{s_{\alpha_i}(\sigma)}}{(\sigma+\epsilon)}$, s_{α_i} is also a learnable linear spline, ϵ is a small non-negative constant ($\epsilon = 1 \cdot 10^{-5}$) and σ corresponds to the noise level. In fact, the authors of [11] proposed to include σ as an input of the regularizer when it is known, otherwise it is set to 1. Hence, the training is done by varying also the noise level across the iterations (see more details in (section 3.3, [11])). To be fair with the other regularizers, we will not use the knowledge of the noise level in the denoising procedure.

In fact, all of the WCRRNN model trained are 1-weakly convex in order to keep the convexity of the following problem:

$$x^* = \arg\min_{x\in\mathbb{R}^{n\times m}} \frac{1}{2}\|x-y\|_2^2 + R^{W}_{\theta}(x). \tag{59}$$

5.2.1 Reconstruction method

Since the WCRRNN models are differentiable, we should solve the relative inverse problem using a gradient based method. We note that the WCRRNN models are trained such that the following objective function is convex:

$$\frac{1}{2}\|x-y\|_2^2 + R^{W}_{\theta}(x).$$

However, the convexity is not guaranteed if we add a hyperparameter $\lambda > 1$ in front of the regularizer, i.e. we consider λR_θ^W instead of R_θ^W. The weak convexity property still holds, which allows us to use the gradient descent algorithm. The authors proposed to use a variant of the accelerated gradient descent, the *SAGD* algorithm (Section 4.1, [11]), which is tailored for weakly convex functions with L-Lipschitz continuous gradient.

5.3 Reconstruction method

We propose here to solve again using the ADMM algorithm in a similar way to section 2.3 even if the convexity of the infimal convolution used does not hold. To counteract the non-convexity, we choose a larger numerical value for the Lagrange hyperparameter as usual. Another approach to solve this inverse problem is the primal/dual approach from [17] which guarantees the convergence to a critical point.

5.4 Numerical result

As already mentionned above, it is very hard to control the strong convexity of R_{TV}^μ via the parameter μ. Moreover, the latter should be as small as possible to have a convenient approximation of the TV regularizer. Therefore, we set $\mu = 1$, since it seems to perform satisfying enough denoising when used alone. Concerning the pretrained WCRRNN model, we chose it to be 1-weakly convex to follow the R_{CNC} framework. However, to make the denoising trough this model performing, we need to add a hyperparameter λ. Hence, we can not guarantee that the final convolution is C-weakly convex with a non-negative constant C.

As for the other numerical experiments, we study the performance of the infimal convolution on the denoising task over the BSD68 dataset with a standard deviation of $\sigma = \frac{10}{255}$.

Performances of the weakly convex regularizers		
Regularizer	psnr	ssim
WCRRNN	23.47	**0.86**
strong approx. of the TV, $\mu = 1$	**27.85**	0.80
Inf. Conv.	27	0.82

Table 4: We compare here the performances of the pretrained WCRRNN model, the 1-strongly convex approximation of the TV regularizer and the infimal convolution of both of them. The infimal convolution regularizer has not the worst psnr or ssim score.

We present here some visual results of the denoising task using each regularizer of table (4).

Figure 10: *Top*:*Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$.*Right*: The denoised image via the pretrained WCRRNN model. *Bottom*:*Left*: The clean image with strides. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$.*Right*: The denoised image via the pretrained WCRRNN model.

Figure 11: *Top*:*Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$.*Right*: The denoised image via the 0.5-strongly convex approximation of the TV regularizer. *Bottom*:*Left*: The clean image with strides. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$.*Right*: The denoised image via the 0.5-strongly convex approximation of the TV regularizer.

Figure 12: *Top*:*Left*: The clean "Lenna" image. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$.*Right*: The denoised image via the infimal convolution regularizer between the pretrained WCRRNN model and the strongly convex approximation of the TV regularizer. *Bottom*:*Left*: The clean image with strides. *Middle*: The noisy image with a noise of $\sigma = \frac{30}{255}$.*Right*: The denoised image via the infimal convolution regularizer between the pretrained WCRRNN model and the strongly convex approximation of the TV regularizer.

6 Conclusion

To conclude this work, we have studied the convex and weakly convex infimal convolution regularizers framework between the TV regularizer (or a strongly convex approximation of it) and a deep learning based regularizer, either the CRRNN or the WCRRNN. Through numerical experiments, we observed that the use of implicit layers approach can perform better than the t-unfolding gradient descent on the CRRNN models. Then, we used these both training approach to construct our own training procedure to learn a CRRNN models under the convex infimal convolution framework, which have the same performances than the convex infimal convolution using pretrained CRRNN models. Finally, we moved on the weakly convex case by studying theoretical approach to construct weakly convex infimal convolution regularizer and observed that it behaves well in practice also. To go further, we could focus on the interpretability of such infimal convolution regularizers by trying to understand how the split ($x = x - y_x + y_x \in \mathbb{R}^{n \times m}$) is made for example.

Appendix A

Proposition A.1. *Let $f : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ be $\frac{\mu}{2}$-weakly convex and $g : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ be μ-strongly convex with $\mu > 0$. If the infimal convolution $f\square g$ is exact, i.e. for all $x \in \mathbb{R}^{n\times m}$, there exists $y_x \in \mathbb{R}^{n\times m}$ such that:*

$$f\square g(x) = \inf_{y\in\mathbb{R}^{n\times m}}\big(f(x-y) + g(y)\big) \tag{60}$$

$$= f(x-y_x) + g(y_x). \tag{61}$$

Then, $f\square g$ is μ-weakly convex.

Proof. We show that $f\square g(\cdot) + \frac{\mu}{2}\|\cdot\|_2^2$ is a convex function. Let $x, x' \in \mathbb{R}^{n\times m}$ and $\lambda \in [0,1]$, $\lambda' = 1-\lambda$.

$$f\square g(\lambda x + \lambda' x') + \frac{\mu}{2}\|\lambda x + \lambda' x'\|_2^2 \tag{62}$$

$$= f(\lambda x + \lambda' x' - y_{\lambda x+\lambda' x'}) + g(y_{\lambda x+\lambda' x'}) + \frac{\mu}{2}\|\lambda x + \lambda' x'\|_2^2 \tag{63}$$

$$\leq f(\lambda x + \lambda' x' - (\lambda y_x + \lambda' y_{x'})) + g(\lambda y_x + \lambda' y_{x'}) + \frac{\mu}{2}\|\lambda x + \lambda' x'\|_2^2 \tag{64}$$

$$= f(\lambda(x-y_x) + \lambda'(x'-y_{x'})) + \frac{\mu}{2}(\|\lambda y_x + \lambda' y_{x'}\|_2^2 + \|\lambda x + \lambda' x'\|_2^2) + g(\lambda y_x + \lambda' y_{x'}) - \frac{\mu}{2}\|\lambda y_x + \lambda' y_{x'}\|_2^2 \tag{65}$$

$$= \Big(f(\lambda(x-y_x) + \lambda'(x'-y_{x'}) + \frac{\mu}{4}\|\lambda(x-y_x) + \lambda'(x'-y_{x'}\|_2^2\Big) + \Big(\frac{\mu}{4}\|\lambda(x+y_x) + \lambda'(x'+y_{x'})\|_2^2\Big) \tag{66}$$

$$+ \Big(g(\lambda y_x + \lambda' y_{x'}) - \frac{\mu}{2}\|\lambda y_x + \lambda' y_{x'}\|_2^2\Big) \tag{67}$$

$$\leq \Big(\lambda(f(x-y_x) + \frac{\mu}{4}\|x-y_x\|_2^2 + \lambda'(f(x'-y_{x'}) + \frac{\mu}{4}\|x'-y_{x'}\|_2^2)\Big) + \Big(\lambda\frac{\mu}{4}\|x+y_x\|_2^2 + \lambda'\frac{\mu}{4}\|x'+y_{x'}\|_2^2\Big) \tag{68}$$

$$+ \Big(\lambda(g(y_x) - \frac{\mu}{2}\|y_x\|_2^2) + \lambda'(g(y_{x'}) - \frac{\mu}{2}\|y_{x'}\|_2^2)\Big) \tag{69}$$

$$= \lambda(f\square g(x) + \frac{\mu}{2}\|x\|_2^2) + \lambda'(f\square g(x') + \frac{\mu}{2}\|x'\|_2^2) \tag{70}$$

We use the fact that this infimal convolution is exact to obtain (63). Then, by definition of the infimal convolution, we can state the inequality (64). After that, we simply use the parallelogram law to obtain (67). To get (69), we use the convexity of the three functions in brackets since f is $\frac{\mu}{2}$-weakly convex and g is μ-strongly convex. Finally, we obtain the last line by using again the parallelogram law, which shows the wanted convexity. □

Appendix B

We show here that $R_{TV}^{\mu}(x) = \sqrt{\|\nabla x\|_2^2 + \mu}$ is C_μ-strongly convex over the subset of discrete representation of images $A \subset \mathbb{R}^{n\times m}$, with C_μ a non-negative constant. To do so, we first show that the function $x \mapsto \sqrt{\|x\|_2^2 + \mu}$ is a C_μ-strongly convex function with a constant $C_\mu > 0$. Then, we show that the strong convexity is preserved with the following transformation $x \mapsto \sqrt{\|\nabla x\|_2^2 + \mu}$.

Proposition B.1. *Let $\mu > 0$, the function $x \mapsto \sqrt{\|x\|_2^2 + \mu}$ is C_μ-strongly convex function with a constant $C_\mu > 0$.*

Proof. We give here only a sketch of the proof. We first need to show that:

$$\sqrt{\|\lambda x + \lambda' x'\|_2^2 + \mu} - \Big(\lambda\sqrt{\|x\|_2^2 + \mu} + \lambda'\sqrt{\|x'\|_2^2 + \mu}\Big) \leq -\frac{C_\mu}{2}\lambda\lambda'\|x - x'\|_2^2 \tag{71}$$

, for all $x, x' \in \mathbb{R}^{n\times m}$ and $\lambda \in [0,1]$, $\lambda' = 1-\lambda$ and

$$C_\mu \leq \min_{x,x'\in A}\left\{\frac{2\mu}{x\cdot x' + \mu + \sqrt{(\|x\|_2^2+\mu)(\|x'\|_2^2+\mu)}}\,\frac{1}{\sqrt{\|\lambda x + \lambda' x'\|_2^2 + \mu} + \lambda\sqrt{\|x\|_2^2+\mu} + \lambda'\sqrt{\|x'\|_2^2+\mu}}\right\}.$$

Since A is a bounded subspace, C_μ is also bounded. To show the above result, we need to combine Young inequalities and the following remarkable identity:

$$\begin{aligned} a^2 - b^2 &= (a-b)(a+b) \\ \iff (a-b) &= \frac{a^2-b^2}{a+b} \end{aligned}$$

□

Proposition B.2. *Let $F : \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}$ be a C-strongly convex function with $C > 0$. Then, $F(\nabla\cdot)$ is also a C_F-strongly convex function over the subset of discrete representation of images$A \subset \mathbb{R}^{n\times m}$ with C_F.*

Proof. First, we notice there is a constant $0 < c_F < \infty$ s.t.

$$c_F\|x\|_2^2 \leq \|\nabla x\|_2^2,$$

for all $x \in (A - A)$ since ∇ is a linear operator on the bounded subset $A - A$. Hence,

$$\begin{aligned} F(\nabla(\lambda x + \lambda' x')) &\leq \lambda F(\nabla x) + \lambda' F(\nabla x') - \frac{C}{2}\lambda\lambda'\|\nabla(x - x')\|_2^2 \\ &\leq \lambda F(\nabla x) + \lambda' F(\nabla x') - \frac{Cc_F}{2}\lambda\lambda'\|x - x'\|_2^2, \end{aligned}$$

for all $x, x' \in A$ and $\lambda \in [0, 1]$, $\lambda' = 1 - \lambda$. So, we set $C_F = C \cdot c_F$. □

References

[1] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, chapter 12. Springer, 2010.

[2] Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

[3] Marta M Betcke and Carola-Bibiane Schönlieb. Mathematics of biomedical imaging today—a perspective, 2023.

[4] Pakshal Bohra, Joaquim Campos, Harshit Gupta, Shayan Aziznejad, and Michael Unser. Learning activation functions in deep (spline) neural networks. *IEEE Open Journal of Signal Processing*, 1:295–309, 2020.

[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 01 2011.

[6] Martin Burger and Stanley Osher. *A Guide to the TV Zoo*, pages 1–70. Springer International Publishing, Cham, 2013.

[7] Piermarco Cannarsa and Carlo Sinestrari. *Semiconcave Functions, Hamilton–Jacobi Equations, and Optimal Control*. Birkhäuser Boston, MA, 2004.

[8] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 2004.

[9] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. *An Introduction to Total Variation for Image Analysis*, pages 263–340. De Gruyter, Berlin, New York, 2010.

[10] Alexis Goujon, Sebastian Neumayer, Pakshal Bohra, Stanislas Ducotterd, and Michael Unser. A neural-network-based convex regularizer for inverse problems, 2023.

[11] Alexis Goujon, Sebastian Neumayer, and Michael Unser. Learning weakly convex regularizers for convergent image-reconstruction algorithms, 2023.

[12] Zico Kolter, David Duvenaud, and Matt Johnson. Deep implicit layers - neural odes, deep equilibirum models, and beyond, 2020.

[13] Yura Malitsky and Konstantin Mishchenko. Adaptive gradient descent without descent, 2020.

[14] Gregory Ongie, Ajil Jalal, Christopher A. Metzler, Richard G. Baraniuk, Alexandros G. Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging, 2020.

[15] Bolin Pan and Marta Betcke. Infimal convolution of curvelet sparsity and total variation for photoacoustic reconstruction, 2023.

[16] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[17] Zakhar Shumaylov, Jeremy Budd, Subhadip Mukherjee, and Carola-Bibiane Schönlieb. Weakly convex regularisers for inverse problems: Convergence of critical points and primal-dual optimisation, 2024.