MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Deep Proximal Gradient Method for Learned Convex Regularizers

Berk, Aaron; Ma, Yanting; Boufounos, Petros T.; Wang, Pu; Mansour, Hassan

TR2023-032 May 06, 2023

Abstract

We consider the problem of simultaneously learning a convex penalty function and its proximity operator for image reconstruction from incomplete measurements. Our goal is to apply Accelerated Proximal Gradient Method (APGM) using a learned proximity op- erator in place of the true proximity operator of the learned penalty function. Starting from a Gaussian image denoiser, we learn an associated penalty function and its proximity operator. The learned penalty function offers provable reconstruction guarantees, whereas access to its proximity operator presents the opportunity to achieve APGM convergence rates, which are faster than those of subgradient descent approaches.

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2023

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

DEEP PROXIMAL GRADIENT METHOD FOR LEARNED CONVEX REGULARIZERS

Aaron Berk[1]*, Yanting Ma*[2]*, Petros Boufounos*[2]*, Pu Wang*[2]*, Hassan Mansour*[2]

[1]Department of Mathematics and Statistics, McGill University, Montréal, QC
[2]Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02140, USA
aaron.berk@mcgill.ca, {yma, petrosb, pwang, mansour}@merl.com

ABSTRACT

We consider the problem of simultaneously learning a convex penalty function and its proximity operator for image reconstruction from incomplete measurements. Our goal is to apply Accelerated Proximal Gradient Method (APGM) using a learned proximity operator in place of the true proximity operator of the learned penalty function. Starting from a Gaussian image denoiser, we learn an associated penalty function and its proximity operator. The learned penalty function offers provable reconstruction guarantees, whereas access to its proximity operator presents the opportunity to achieve APGM convergence rates, which are faster than those of subgradient descent approaches.

***Index Terms*—** Deep learning, proximal gradient method, plug-and-play prior, input-convex neural network

1. INTRODUCTION

Linear inverse problems are an important class of problems within the realm of computational imaging [1–3]. Suppose we wish to estimate an unknown image x_* from measurements $b = Ax_* + \xi$ where A is the measurement matrix (forward operator) and ξ is noise with variance σ^2. The arguably *de facto* convex optimization approach for solving such a problem is to estimate x_* as

$$\hat{x} := \operatorname*{argmin}_{x \in \mathbb{R}^n} \ell(x; b) + \lambda\rho(x), \tag{1}$$

where $\ell(x; b) := \frac{1}{2\sigma^2}\|Ax - b\|_2^2$ is the loss function, ρ is a convex regularizer suitably adapted to the natural structure of the image class, and $\lambda > 0$ is a regularization parameter. One effective approach for computing $\hat{x}$ is the (accelerated) proximal gradient method (APGM). For (1) this takes the form

$$\begin{aligned} v_{k+1} &\leftarrow x_k - \mu\nabla\ell(x_k; y) \\ z_{k+1} &\leftarrow \operatorname{prox}(\mu\lambda\rho)(v_{k+1}) \\ x_{k+1} &\leftarrow z_{k+1} + \alpha_k(z_{k+1} - z_k), \end{aligned}$$

where the proximity operator (or prox) of a function $f : \mathbb{R}^n \to \mathbb{R}_\infty$ is defined by

$$\operatorname{prox}_f(x) := \operatorname*{argmin}_{w \in \mathbb{R}^n} \left\{ \frac{1}{2}\|x - w\|_2^2 + f(w) \right\}.$$

In general, $\operatorname{prox}_f(x)$ is a set-valued mapping. However, if $f : \mathbb{R}^n \to \mathbb{R}_\infty$ is lower semicontinuous (lsc) [4, Definition 1.21] and convex [4, Definition 8.1], then $\operatorname{prox}_f(x)$ is single-valued [4, Proposition 12.15] and $\operatorname{Fix}\operatorname{prox}_f = \operatorname{argmin} f$ [4, Proposition 12.28] where $\operatorname{Fix} T = \{x \in \mathbb{R}^n : x = T(x)\}$ denotes the set of fixed points of the map $T : \mathbb{R}^n \to \mathbb{R}^n$.

Inverse problems in computational imaging have benefitted from recent advances in deep learning [1, 4–12]. Image denoisers, including neural networks (NNs), serve implicitly as natural image priors, and may be used to substitute proximity operators in convex minimization algorithms [5]. Recent work [13] has examined the role of learned convex regularizers for solving inverse problems. There, the authors select the convex regularizer $\rho(\cdot)$ to be a learned input-convex neural network (ICNN). Thus, one [13] may first learn a suitable convex regularizer $\rho_{\text{ICNN}} = f_\theta(x)$ for the image class of interest, and use it as the penalty function in the above convex optimization problem (1). This approach admits provable guarantees for a subgradient descent method, deriving from the input-convex assumption on the penalty function.

Unfortunately, such subgradient descent approaches are known to admit poor convergence rates for non-smooth objective functions [14]. Indeed, $\ell + \lambda\rho$ can be non-smooth: NNs commonly incorporate piecewise linear activation functions like ReLU. Thus, it is natural to seek an alternative approach admitting provable guarantees for an algorithm with known faster convergence rates for non-smooth objectives.

1.1. Contributions

In this work we consider the problem of simultaneously learning a convex penalty function $\rho = f_\theta$ and its proximity operator $H_\psi \approx \operatorname{prox}_{f_\theta}$. We apply a version of PnP-APGM using H_ψ in place of prox_ρ. To this end, we initialize H_ψ as a Gaussian denoiser and learn H_ψ while simultaneously learning the associated penalty function. The learned penalty function offers the performance benefits discussed in [13], whereas access to its prox presents the opportunity to achieve APGM convergence rates, which are faster than those of subgradient descent approaches.

To eke out provable guarantees, we assume that the architecture of ρ is an ICNN, $\rho = \rho_{\text{ICNN}} = f_\theta(x) : \mathbb{R}^n \to \mathbb{R}$, and we approximate prox_ρ by a NN $H_\psi : \mathbb{R}^n \to \mathbb{R}^n$. For example, in our numerical experiments H_ψ is a DnCNN [10]. However, it could equally well be a variational autoencoder (VAE) [15] or U-Net [16]. We show, under suitable assumptions on the convergence of the networks during training, that PnP-PGM with H_ψ yields provable convergence guarantees, with faster rates than those achievable by subgradient descent.

1.2. Notation

Throughout, we assume that $x, y \in \mathbb{R}^n \cong \mathbb{R}^{d\times d}$ are vectors that represent square images. We may refer to them as "signals". The "square" assumption is for notational convenience; it

A. Berk conducted this work partially during a summer internship at MERL. They are currently partially supported by Institut de valorisation des données (IVADO) and Centre de Recherches en Mathématiques (CRM) Applied Math Lab.

can be removed without impacting our results. Denote the extended reals by $\mathbb{R}_\infty = \mathbb{R} \cup \{+\infty\}$ and the nonnegative reals by $\mathbb{R}_+ := \{x \in \mathbb{R} : x \geq 0\}$. A matrix B is positive definite if $B \succ 0$.

2. RELATED WORK

Regularization by denoising (RED) and Plug-and-Play priors are two related approaches that have demonstrated the feasibility of incorporating deep image priors in iterative reconstruction algorithms. In regularization by denoising, general-purpose signal denoisers yield explicit regularizers for inverse problems [6, 7]. In the plug-and-play (PnP) priors approach [8], ill-posed inverse problems are solved by an iterative proximal-type algorithm (like APGM or ADMM), in which a prox is swapped with a (typically non-convex) denoising operation. For example, a PnP variant of APGM might replace the prox with BM3D [9], DnCNN [10] or a firmly non-expansive ([4, Definition 4.1]) convolutional neural network (CNN) [11]. One commonly substitutes the prox for a Gaussian denoiser when implementing PnP methods in practice [9, 10, 12]. In such cases, it is empirically well supported that the so-called PnP-APGM and PnP-ADMM approaches obtain good recovery results [8, 12, 17].

ICNNs were proposed as tools for inference and optimization tasks in [18]. In simplest terms, a d-layer ICNN $z_d = f_\theta(x)$ takes the form

$$z_{i+1} := g_i(W_i^{(z)} z_i + W_i^{(x)} x + b_i), i = 0, \ldots, d-1.$$

Above, z_i denotes the layer activations with $z_0, W_0^{(z)} = 0$; g_i the activation functions for each layer; and $\theta := (W_{0:k-1}^{(x)}, W_{1:k-1}^{(z)}, b_{0:k-1})$. If $W_{1:k-1}^{(z)}$ are non-negative and all g_i are convex and non-decreasing, then the ICNN f is convex in x [18, Proposition 1]. Note CNNs can be made input-convex since convolutions are linear transformations.

Convergence rates and computational cost for vanilla proximal gradient methods can be found in [14, Ch. 10], under mild regularity assumptions when ρ is closed and convex; [14, Theorem 8.13] for projected subgradient descent. Previous work has analyzed convergence of inexact proximal gradient methods, in which there is error in both the computation of the gradient and the proximal mapping [19]. There, the authors required that the sequence of errors satisfy certain summability conditions to establish sublinear and linear convergence rate guarantees. Subsequent work [20] has analyzed the non-convex non-smooth setting, using an approach that still requires summability of the noise terms. From a practitioner's standpoint, this summability assumption can be read as a requirement of computation to a pre-specified accuracy. However, such an assumption may be unrealistic for pre-computed functions designed to approximate proximity operators.

Convergence guarantees for some PnP methods using a gradient-step denoiser, which corresponds with the proximal operator of some scalar function (by [21]), appear in [22]. See [23] for PnP convergence guarantees in a compressed sensing setting. In [12], convergence guarantees for PnP-PGM are given for a class of denoisers satisfying a nonexpansivity condition.

3. THEORY

Suppose $\rho : \mathbb{R}^n \to \mathbb{R}_+$ in (1) is an unknown target penalty function. Let $f_\theta : \mathbb{R}^n \to \mathbb{R}_+$ be an ICNN parametrized by $\theta \in \mathbb{R}^p$ with $f_\theta(x)$ lsc in x for each θ, and let $H_\psi : \mathbb{R}^n \to \mathbb{R}^n$ be a NN with parameters $\psi \in \mathbb{R}^q$. Next, suppose $\mathcal{D}$ is a data distribution of interest with marginals $\mathcal{D}_c$ and $\mathcal{D}_n$ representing "clean" and "noisy" images, respectively. To obtain a penalty function $f_{\hat{\theta}}$ and its approximate prox $H_{\hat{\psi}}$, we propose jointly learning $(\hat{\theta}, \hat{\psi}) \in \mathbb{R}^{p \times q}$ to solve the saddle optimization

$$\min_{\psi \in \mathbb{R}^q} \max_{\theta \in \mathbb{R}^p} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big\{ \frac{1}{2} \|H_\psi(y) - y\|_2^2 + f_\theta(H_\psi(y)) + f_\theta(y) - 2f_\theta(x) \Big\}. \tag{2}$$

We believe our formulation (2) to be novel. Note (2) resembles the generative adversarial network (GAN) optimization problem [24, p.58]: the penalty function f_θ is analogous to the *discriminator*, the prox H_ψ, the *generator*. Furthermore, in the objective, f_θ aims to maximize the term $f_\theta(H_\psi(y))$, whereas H_ψ aims to minimize it.

We now motivate the design of this program. First, if $\hat{\theta}$ were known *a priori*, then (2) is equivalent to

$$\min_{\psi \in \mathbb{R}^q} \mathop{\mathbb{E}}_{y\sim\mathcal{D}_n} \left\{ \frac{1}{2} \|H_\psi(y) - y\|_2^2 + f_{\hat{\theta}}(H_\psi(y)) \right\}. \tag{3}$$

If H_ψ were sufficiently *expressive*, then (3) promotes learning $\hat{\psi}$ such that $H_{\hat{\psi}}$ acts like $\text{prox}_{f_{\hat{\theta}}}$, at least for $y \sim \mathcal{D}_n$. Indeed, one might hope for $\hat{\psi}$ solving (3) and $\varepsilon_H > 0$ sufficiently small with

$$\mathbb{E}_{y\sim\mathcal{D}_n} \|H_{\hat{\psi}}(y) - \text{prox}_{f_{\hat{\theta}}}(y)\|_2 \leq \varepsilon_H.$$

Quantifying the expressivity of H_ψ is beyond the scope of this paper; we support the efficacy of our approach empirically in §4.

On the other hand, if $\hat{\psi}$ were known, then (2) is equivalent to

$$\min_{\theta \in \mathbb{R}^p} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left\{ f_\theta(x) - \frac{1}{2} \left(f_\theta(y) + f_\theta(H_{\hat{\psi}}(y)) \right) \right\}. \tag{4}$$

Observe that (4) promotes a penalty f_θ that assigns low scores to "clean" images $x \sim \mathcal{D}_c$, and high scores to "noisy" or "artificial" images $y, H_{\hat{\psi}}(y)$, where $y \sim \mathcal{D}_n$. This idea is consistent with the denoising-based approaches discussed in §1.

3.1. Algorithm

We next present our main numerical procedure (5), which is obtained by using $H_{\hat{\psi}}$ with PnP-APGM. We then provide a convergence guarantee for our method under a regularity assumption on $H_{\hat{\psi}}$.

Suppose $(\hat{\theta}, \hat{\psi}) \in \mathbb{R}^{p \times q}$ solves (2). Motivated by PnP-APGM, with the idea that $H_{\hat{\psi}}$ reasonably approximates $\text{prox}_{f_{\hat{\theta}}}$, we define the accelerated deep proximal gradient (ADPG) iteration for each integer $t \geq 0$:

$$\begin{aligned} x^{(t+1/2)} &:= T_{\hat{\psi}}(x^{(t)}) \\ x^{(t+1)} &:= x^{(t+1)} + \beta_t \left(x^{(t+1/2)} - x^{(t)} \right) \\ T_{\hat{\psi}}(x^{(t)}) &:= H_{\hat{\psi}} \left(x^{(t)} - \alpha A^*(Ax^{(t)} - b) \right). \end{aligned} \tag{5}$$

Note that $x^{(0)} \in \mathbb{R}^n$ is a chosen initialization (*e.g.,* one may take $x^{(0)} := H_{\hat{\psi}}(A^*b)$ in practice). When $\beta_t = 0$, (5) converges to $\text{Fix}(T_{\hat{\psi}})$. Importantly, it is reasonable to expect that $\text{Fix}(T_{\hat{\psi}}) \neq \emptyset$ (*e.g.,* see [4, Theorem 4.19]).

Theorem 3.1. *Let $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ and $A^*A \succ 0$. Let $b := Ax_* + \xi$ for $x_* \in \mathbb{R}^n$ and $\xi \in \mathbb{C}^m$. With $f_\theta : \mathbb{R}^n \to \mathbb{R}$ parametrized by $\theta \in \mathbb{R}^p$ such that f_θ is convex in its argument for*

each $\theta \in \mathbb{R}^p$, and $H_\psi : \mathbb{R}^n \to \mathbb{R}^n$ parametrized by $\psi \in \mathbb{R}^q$, let $(\hat{\theta}, \hat{\psi})$ solve (2). Let $\hat{x} \in \mathbb{R}^n$ solve

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|b - Ax\|_2^2 + \frac{1}{\alpha} f_{\hat{\theta}}(x),$$

where $0 < \alpha < \lambda_{\max}^{-1}(A^*A)$. If $H_{\hat{\psi}}$ is nonexpansive and $\bar{x} \in \mathrm{Fix}(T_{\hat{\psi}})$, then there exists $\nu \in (0,1)$ and $\varepsilon = \varepsilon(H_{\hat{\psi}}, \hat{x}) > 0$ so that for each integer $t \geq 0$, the ADPG iterates (5) with $\beta_t = 0$ satisfy

$$\|x^{(t)} - \hat{x}\|_2 \leq \nu^t \|x^{(0)} - \bar{x}\|_2 + \frac{\varepsilon}{1-\nu}.$$

The proof of the theorem is standard (*e.g.,* see [23]) and is omitted due to space limitation.

4. NUMERICAL EVALUATION

First, we establish the general framework of our numerical approach. We then specify additional implementation details (§4.1) and showcase our algorithm on a specific numerical example (§ 4.2). Some implementation details specific to § 4.2 were used to more rapdily obtain a working algorithm with greater performance and robustness; we expand on this below.

Our numerics were implemented in PyTorch [25]. We implement the optimization of (2) via an alternating minimization procedure, the two objectives being (3) and (4). We outline the structure of the alternating minimization procedure in PyTorch-flavoured pseudo-code in Algorithm 4.1. In lines 3–5, a batch of clean and noisy data is sampled from the training distribution; f_{θ_i} is given by `f` and H_{ψ_i} by `H`. Lines 7–8 give the batch loss for the objective (4); lines 12–13 for (3). In lines 9–10, the weights of `H` are frozen and `.backward` computes the gradient of the loss with respect to the weights of `f`. Then, the optimizer `optimizer_penalty` updates the weights of the penalty function using the just-computed gradient. Similarly, in lines 14–15 the weights of `f` are frozen, the gradient of the loss is computed with respect to the weights of `H`, and `optimizer_prox` updates the weights of `H` using this gradient.

```
1  for epoch in range(num_epochs):
2      for batch in train_distr:
3          x, y = batch
4          fx, fy, Hy = f(x), f(y), H(y)
5          fHy = f(Hy)
6          # stage 1
7          loss_penalty = mean(
8              fx - 0.5 * (fy + fHy))
9          loss_penalty.backward()
10         optimizer_penalty.step()
11         # stage 2
12         loss_prox = (
13             mse(Hy, y) + λ * mean(f(Hy)))
14         loss_prox.backward()
15         optimizer_prox.step()
```

Algorithm 4.1: Outline of the alternating minimization procedure in PyTorch-flavoured pseudo-code.

4.1. Implementation details

Because (2) is a saddle point problem, we implemented the alternating minimization procedure using Optimistic Adam [26, 27]. On lines 10, 15 of Algorithm 4.1, `optimizer_penalty` and `optimizer_prox` are functions implementing an iteration of Optimistic Adam for θ and ψ, respectively.

In addition to the loss computed for f_θ above, we found it beneficial to regularize θ to promote 1-Lipschitzness of f_θ, using the spectral norm regularization approach developed in [28]. We adapted this approach to ICNNs by changing the Lipschitz target constants for $W_i^{(z)}, W_i^{(x)}$ from 1 to $\frac{1}{\sqrt{2}}$. Thus, to `loss_penalty` we added a spectral regularization quantity, multiplied by a large regularization constant, 10^4.

Since f_θ is an unknown penalty function, we initialized its weights using default PyTorch settings. However, it was helpful to select a good initialization for ψ. To this end, we use a transfer learning approach [29] to initialize the weights of the prox function H_ψ using the trained RealSN-DnCNN developed in [12], which we denote by J_σ. During the alternating minimization procedure we use the same training and validation images as [12] to form the clean and noisy data distributions, namely the BSD500 dataset [30] images divided into 40×40 patches.

4.2. Compressed sensing MRI

We perform a compressed sensing magnetic resonance imaging (CS-MRI) experiment. We use the setup of [12, Section 6] so as to be able to draw a direct comparison with their approach. Define the measurement process

$$b = Ax_{\text{true}} + \xi,$$

where $x_{\text{true}} \in \mathbb{C}^n$ is the underlying image, $A \in \mathbb{C}^{m \times n}$ is the linear measurement model and $b \in \mathbb{C}^m$ are the measured data. Above, $\xi \in \mathbb{C}^m$ is some unknown (possibly random) corruption. In our experiments, x_{true} will be one of two images, to which we refer as "Brain" and "Bust", respectively (images on GitHub).

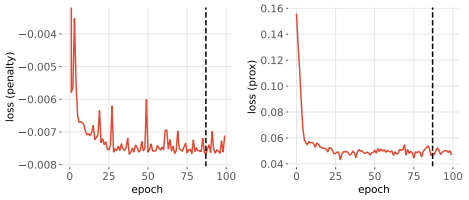
As in [12], A is a partial Fourier operator (subsampled discrete Fourier transform), with random, radial or Cartesian sub-sampling at a sampling rate of 30%. See [31] for more information. Following [12], the noise ξ was sampled from a complex normal distribution with $\mathrm{Re}(\xi), \mathrm{Im}(\xi) \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma)$, $\sigma = 15/255$.

Our aim is to solve the following minimization problem

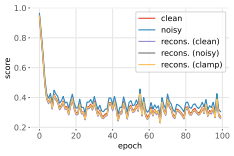
$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|b - Ax\|_2^2 + \lambda f_{\hat{\theta}}(x),$$

where $\lambda := 10^{-2}\sigma$ (unless specified otherwise). For an image $y \in \mathbb{C}^{d \times d}$ with $d^2 = n$, $J_\sigma(y)$ outputs the "noise" in the image y, so we initialize H_ψ as $H_\psi(x) := x - J_\sigma(x)$, where ψ denotes the weights of J_σ, which are tuned during training. The penalty function that we train is an ICNN [18]. Specifically, it is an input-convex CNN with hidden layer sizes $[16, 32, 64]$. In particular, all convolutions are 3×3 convolutions with stride 1 and padding 1, $(W^{(0)}, W^{(1)}, W^{(2)})$ have $16, 32, 64$ filters respectively, $(\tilde{W}^{(1)}, \tilde{W}^{(2)})$ have $32, 64$ filters respectively, and the biases are given by $b^{(i)}$ for $i = 0, 1, 2, 3$. Thus, the penalty function $f_\theta(x)$ takes the form

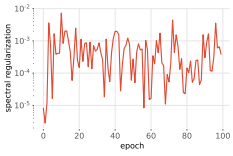
$$\begin{aligned} y^{(1)} &= \mathrm{LReLU}(\mathrm{mp}(W^{(0)}x + b^{(0)})) \\ y^{(j+1)} &= \mathrm{LReLU}(\mathrm{mp}(\mathrm{aap}(W^{(j)}x) + \tilde{W}^{(j)}y^{(j-1)} + b^{(j)})) \\ f_\theta(x) &= \mathrm{ReLU}(W^{(3)}\,\mathrm{aap}(y^{(3)}) + b^{(3)}) \end{aligned}$$



(a) Validation loss for penalty (left) and prox (right).



(b) Penalty loss per image type.



(c) Spectral regularization loss.

Fig. 1: Per-epoch validation metrics for the alternating optimization procedure described in Algorithm 4.1.

for $j = 1, 2$, and where the final linear mapping $W^{(3)}$ denotes a fully connected layer with scalar output. Above, LReLU denotes the leaky ReLU function with slope 10^{-2}, ReLU denotes the ReLU function, mp denotes 2×2 max pooling, and aap denotes 2D adaptive average pooling. Note the final adaptive average pooling has output size 1, while the output size of the others is determined by the spatial dimensions of $y^{(j-1)}$. We refer the reader to the PyTorch documentation [25] for further implementation details pertaining to the above functions and their optimization.

The networks were optimized with Optimistic Adam [26, Algorithm 1] with learning rates $10^{-3}, 10^{-4}$ for f_θ, H_ψ, respectively, and a weight decay of 10^{-4} in both cases. Training was run for 100 epochs with batch size 128 and the model with greatest validation peak signal-to-noise ratio (PSNR) was selected (epoch 87). Training results (*i.e.,* per-epoch validation metrics) appear in Figure 1.

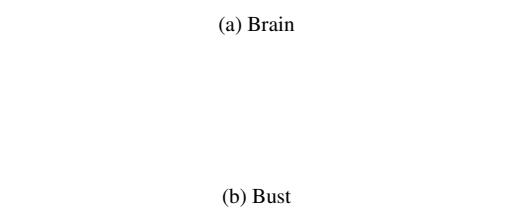
Next, we run ADPG as in (5) with our trained $H_{\hat{\psi}}$ with no acceleration (*i.e.,* $\beta_t = 0$) for a maximum of 25 iterations (we found that changing β_t did not significantly impact rate of convergence or image quality). We evaluate success of recovery on the "Brain" and "Bust" images using PSNR. We compare final PSNR values for ADPG to subgradient descent with $\lambda = 100\sigma$ (using $\nabla_x f_{\hat{\theta}}(x)$) and to PnP-FBS [12]. As a benchmark, we also compute the PSNR values for $\hat{x}_{\text{FBP}}$ and $H_{\hat{\psi}}(\hat{x}_{\text{FBP}})$. We call $H_{\hat{\psi}}(\hat{x}_{\text{FBP}})$ the "denoised" image, by which we mean the image recovered by a single application of $H_{\hat{\psi}}$ to the noisy image: $H(\hat{x}_{\text{FBP}}; \psi)$, where $\hat{x}_{\text{FBP}} := A^\dagger b$ is the (zero-filled) "filtered back-projection" of the noisy measurements. The five PSNR values for each image and sampling mask appear in Table 1. In each row, the greatest PSNR value has been bolded. In Figure 2 we show (from left-to-right) the Brain and Bust images, the recovered images using PnP-FBS, ADPG and $H_{\hat{\psi}}(\hat{x}_{\text{FBP}})$), and the noisy image $\hat{x}_{\text{FBP}}$ when the sampling mask was Random.

Finally, in Figure 3 we plot PSNR as a function of iteration number for both ADPG (solid line) and subgradient descent (dashed line). For visual clarity we log-transform the horizontal axis. Each plot corresponds to a pairing of an image and a sampling mask, as determined by the row and column labels. It is easily seen that ADPG requires substantially fewer iterations to reach its maximal PSNR, as compared with subgradient descent.

5. CONCLUSION

In this work, we propose a nonconvex optimization program to simultaneously obtain parameters $(\hat{\theta}, \hat{\psi})$ for an ICNN f_θ, which serves as a penalty function for the linear inverse problem (1), and a NN H_ψ which serves as an approximation to $\text{prox}_{f_{\hat{\theta}}}$. We show that our accelerated deep proximal gradient method (ADPG), being PnP-APGM using $H_{\hat{\psi}}$, admits provable convergence guarantees with a regularity assumption on $H_{\hat{\psi}}$ (Theorem 3.1). We showcase how to approximately solve (2) via Algorithm 4.1. Finally, we perform a numerical evaluation that demonstrates competitive recovery PSNR with PnP-FBS and subgradient descent with learned penalty $f_{\hat{\theta}}$.

In future work, we wish to quantify ε in Theorem 3.1 and ε_H from § 3, both theoretically and empirically. Finally, it remains an open question to determine conditions under which $H_{\hat{\psi}}$ is nonexpansive.



(a) Brain

(b) Bust

Fig. 2: Results on Brain and Bust images with Random sampling mask. Left-to-right: true image, PnP-FBS [12], **ADPG**, $H_{\hat{\psi}}(\hat{x}_{\text{FBP}})$, $\hat{x}_{\text{FBP}}$. PSNR values for latter 4 images in rows 3 and 6 of Table 1, respectively.

		$\hat{x}_{\text{FBP}}$	$H_{\hat{\psi}}$	subGD	**ADPG**	PPnP
Brain	Cartesian	20.22	20.80	22.87	22.65	**23.13**
	Radial	21.74	22.45	**26.33**	26.16	25.93
	Random	22.23	23.11	**27.15**	27.08	27.06
Bust	Cartesian	21.74	22.66	25.42	24.77	**25.46**
	Radial	22.76	23.89	**27.77**	27.34	27.40
	Random	23.92	25.35	28.45	28.21	**28.65**

Table 1: PSNR summary for the methods considered in this work. **Columns:** $\hat{x}_{\text{FBP}}$: noisy images $\hat{x}_{\text{FBP}}$; $H_{\hat{\psi}}$: denoised images $H_{\hat{\psi}}(\hat{x}_{\text{FBP}})$; subGD: subgradient descent with $\nabla_x f_{\hat{\theta}}$, $\alpha = 0.1$; ADPG: deep proximal gradient descent with $H_{\hat{\psi}}$, $\alpha = 1$, $\lambda = 100\sigma$; PPnP: PnP-FBS [12] with RealSN-DnCNN, $\alpha = 0.4$.

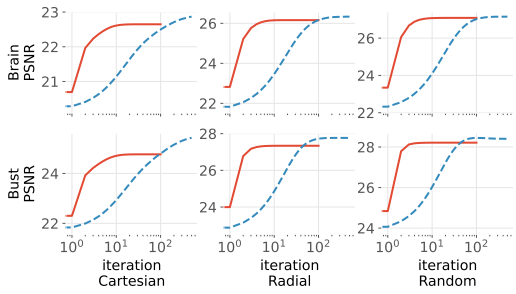


Fig. 3: Plot of PSNR as a function of iteration number for ADPG (solid line) and subgradient descent with $f_{\hat{\theta}}$ (dashed line).

6. REFERENCES

[1] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery," *IEEE Signal Processing Magazine*, vol. 37, no. 1, pp. 105–116, 2020.

[2] A. Ribes and F. Schmitt, "Linear inverse problems in imaging," *IEEE Signal Processing Magazine*, vol. 25, no. 4, pp. 84–99, 2008.

[3] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[4] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011, vol. 408.

[5] T. Meinhardt, M. Moller, C. Hazirbas, and D. Cremers, "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1781–1790.

[6] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1804–1844, 2017.

[7] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Transactions on Computational Imaging*, vol. 5, no. 1, pp. 52–67, 2018.

[8] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 945–948.

[9] W. De Leeuw and R. Van Liere, "BM3D: Motion estimation in time dependent volume data," in *IEEE Visualization, 2002. VIS 2002.* IEEE, 2002, pp. 427–433.

[10] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[11] M. Terris, A. Repetti, J.-C. Pesquet, and Y. Wiaux, "Building firmly nonexpansive convolutional neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8658–8662.

[12] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5546–5557.

[13] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb, "Learned convex regularizers for inverse problems," *arXiv preprint arXiv:2008.02839*, 2020.

[14] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[17] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, C.-B. Schönlieb, and H. Huang, "Tuning-free plug-and-play proximal algorithm for inverse imaging problems," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 158–10 169.

[18] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 146–155.

[19] M. Schmidt, N. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[20] B. Gu, D. Wang, Z. Huo, and H. Huang, "Inexact proximal gradient methods for non-convex and non-smooth optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[21] R. Gribonval, "Should penalized least squares regression be interpreted as maximum a posteriori estimation?" *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2405–2410, 2011.

[22] S. Hurault, A. Leclaire, and N. Papadakis, "Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9483–9505.

[23] J. Liu, S. Asif, B. Wohlberg, and U. Kamilov, "Recovery analysis for plug-and-play priors using the restricted eigenvalue condition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5921–5933, 2021.

[24] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[26] C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng, "Training GANs with optimism," *arXiv preprint arXiv:1711.00141*, 2017.

[27] P. Mertikopoulos, B. Lecouat, H. Zenati, C.-S. Foo, V. Chandrasekhar, and G. Piliouras, "Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile," *arXiv preprint arXiv:1807.02629*, 2018.

[28] S. Singla and S. Feizi, "Fantastic four: Differentiable and efficient bounds on singular values of convolution layers," in *International Conference on Learning Representations*, 2020.

[29] A. Ng, "Nuts and bolts of building AI applications using deep learning," *NIPS Keynote Talk*, 2016.

[30] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2. IEEE, 2001, pp. 416–423.

[31] E. M. Eksioglu, "Decoupled algorithm for MRI reconstruction using nonlocal block matching model: BM3D-MRI," *Journal of Mathematical Imaging and Vision*, vol. 56, no. 3, pp. 430–440, 2016.