Uniformly convex neural networks and non-stationary iterated network Tikhonov (iNETT) method

Davide Bianchi, Guanghao Lai and Wenbin Li

School of Science, Harbin Institute of Technology, Shenzhen, Shenzhen 518055, People’s Republic of China
E-mail: liwenbin@hit.edu.cn

Received 6 October 2022; revised 31 January 2023
Accepted for publication 9 March 2023
Published 27 March 2023

Abstract
We propose a non-stationary iterated network Tikhonov (iNETT) method for the solution of ill-posed inverse problems. The iNETT employs deep neural networks to build a data-driven regularizer, and it avoids the difficult task of estimating the optimal regularization parameter. To achieve the theoretical convergence of iNETT, we introduce uniformly convex neural networks to build the data-driven regularizer. Rigorous theories and detailed algorithms are proposed for the construction of convex and uniformly convex neural networks. In particular, given a general neural network architecture, we prescribe sufficient conditions to achieve a trained neural network which is component-wise convex or uniformly convex; moreover, we provide concrete examples of realizing convexity and uniform convexity in the modern U-net architecture. With the tools of convex and uniformly convex neural networks, the iNETT algorithm is developed and a rigorous convergence analysis is provided. Lastly, we show applications of the iNETT algorithm in 2D computerized tomography, where numerical examples illustrate the efficacy of the proposed algorithm.

Keywords: iterated network Tikhonov, uniformly convex neural networks, data-driven regularizer, U-net, regularization of inverse problem

(Some figures may appear in colour only in the online journal)
1. Introduction

Consider the discretized form of an ill-posed linear problem,
\[ F x = y, \]  
(1.1)
where \( X \) and \( Y \) denote finite dimensional normed spaces, i.e. \( X = (\mathbb{R}^N, \| \cdot \|_X) \), \( Y = (\mathbb{R}^M, \| \cdot \|_Y) \), and \( F : X \rightarrow Y \) is the discretization of an ill-posed linear operator \( F \). The inverse problem aims to recover \( x \) from the observed data \( y^\delta \) contaminated by unknown error with bounded norm,
\[ y^\delta = y + \eta, \quad \text{where} \quad \| \eta \|_Y \leq \delta. \]
As \( y^\delta \) is not necessarily in the range of \( F \), i.e. \( y^\delta \notin \text{Rg}(F) \), we consider a variational approach to solve the inverse problem,
\[ x^\delta := \arg\min_{x \in X} \| Fx - y^\delta \|_Y^2. \]  
(1.2)

For the ill-posed inverse problem, regularization techniques should be introduced when solving equation (1.2). The regularization aims to provide prior knowledge and improve stability of the solution. For example, a typical choice of regularization in imaging is the \( \ell^p \)-norm of \( x \), with \( p \geq 1 \), which can be weighted by the Laplacian operator with Dirichlet or Neumann boundary condition \([3, 23]\). Recently, deep learning approaches are introduced to develop data-driven regularization terms in the solutions of inverse problems. In \([37]\), the authors propose a network Tikhonov (NETT) approach, which combines deep neural networks with a Tikhonov regularization strategy. The general form of NETT can be summarized as follows,
\[ x^\delta := \arg\min_{x \in \text{dom}(F) \cap \text{dom}(\Phi \Theta)} A(Fx, y^\delta) + \alpha \psi(\Phi \Theta(x)), \]  
(NETT)
where \( A(Fx, y^\delta) \geq 0 \) is the data-fidelity term which measures misfits between the approximated and measurement data, \( \alpha > 0 \) is a regularization parameter, and \( \psi(\Phi \Theta(x)) \) is the regularization (or penalty) term including a neural network architecture \( \Phi \Theta \). In particular, \( \psi \) is a nonnegative functional, and the neural network \( \Phi \Theta \) is trained to penalize artifacts in the recovered solution. By training \( \Phi \Theta \) in an appropriate way, the neural-network based regularization term is able to capture the feature of solution errors due to data noises and the inexact iterative scheme, so that it can provide penalization on the artifacts of solutions in an adaptive manner. This data-driven regularization strategy shows many advantages in solving inverse problems, and related studies can be found in \([4, 6, 43]\) and \([1]\) as well.

Motivated by NETT, we propose an iterated network Tikhonov (iNETT) method which combines the data-driven regularization strategy with an iterated Tikhonov method. In a Tikhonov-like method as NETT, the regularization parameter \( \alpha \) plays an important role since it controls the trade-off between the data-fidelity term and the regularization term. The value of \( \alpha \) relies on the noise level \( \delta \), and it will affect the proximity of the recovered solution to the minimizer of the data-fidelity term. Poor choices of \( \alpha \) can lead to very poor solutions, and it is well known that an accurate estimate of the optimal \( \alpha \) is difficult to achieve and it typically relies on heuristic assumptions (e.g. \([26, 28, 46]\)). As a result, a natural strategy is to consider an iterated Tikhonov method with non-stationary values of \( \alpha \) in the iteration. The non-stationary iterated Tikhonov method is able to avoid exhaustive tuning of the regularization parameter, and it achieves better convergence rates in many applications. For example, we refer the readers to \([12, 13, 16, 18, 20, 22, 27]\) for the applications of iterated Tikhonov in Hilbert spaces, and \([9, 34, 35, 45]\) in Banach spaces.
Combining the strategy of neural-network based regularizer with the non-stationary iterated Tikhonov method, the iNETT method has the following general form,

\[
\begin{align*}
  x_n^\delta &:= \text{argmin}_{x \in X} \frac{1}{2} \|Fx - y^\delta\|_2^2 + \alpha_n B_{\xi_{n-1}}^{R}(x, x_{n-1}), \\
  \xi_n^\delta &:= \xi_{n-1}^\delta - \frac{1}{\alpha_n} F^T J_r (Fx_n^\delta - y^\delta),
\end{align*}
\]

where \( R := \Phi_{\Theta}^m : X \to (\mathbb{R}, |\cdot|) \) is a uniformly convex neural network, \( B_{\xi_{n-1}}^{R}(\cdot, \cdot) \) is the Bregman distance induced by \( R \) in the direction \( \xi_{n-1}^\delta \in \partial R(x_{n-1}) \), \( J_r \) denotes the duality map for \( r \in (1, \infty) \), and \( \{\alpha_n\}_n \) is a sequence of positive real numbers. The value of \( \alpha_n \) controls the amount of regularization, and it plays the role of regularization parameter. By taking a decreasing sequence of \( \{\alpha_n\}_n \) and considering the standard discrepancy principle as the stopping rule, the iNETT algorithm can automatically determines the amount of regularization. We will provide the details of iNETT in section 5, including a rigorous convergence analysis and many implementation details.

In the formula of iNETT, the neural network \( \Phi_{\Theta}^m \) is employed to build the regularization term, and it is required to be uniformly convex. The property of uniform convexity is demanded in the convergence analysis of the iterated Tikhonov method [35]. As a result, another important aspect of the paper is the modeling of convex and uniformly convex neural networks. In section 2, we provide an exact mathematical modeling for the general architecture of neural networks. Our modeling can express the modern convolutional neural networks, where the operations like skip connection and concatenation are included. In section 3, we propose rigorous theories for the convex and uniformly convex neural networks. Given a general neural network \( \Phi_{\Theta} : X \to Z \), we prescribe sufficient conditions to obtain a related neural network which is component-wise convex or uniformly convex. The main idea comes from some recent works on convex neural networks, e.g. [2, 42, 51, 53], but we largely extend them to build modern architectures which can embrace state-of-the-art neural networks. In section 4, we provide particular examples of convex and uniformly convex U-net architectures. The U-net is a convolutional neural network widely used in image processing and related imaging science [48]. We give rigorous formulas for the U-net architecture, and explain the approaches to obtain convex and uniformly convex U-net architectures according to the general theories proposed in section 3. In sections 5 and 6, we provide implementation details as we employ the convex U-net to build a uniformly convex regularizer for the iNETT algorithm. The proposed method is successfully applied to computerized tomography in section 6. The tool of convex and uniformly convex neural networks is actually a by-product when designing the iNETT algorithm, but it seems more interesting than the algorithm itself. The tool of convex neural networks shall have many interesting applications in the future study.

2. Notation and setting

We collect here most of the notations and definitions we will use through this work. As main references, the reader can look at [47, 50, 56]. First of all, let us fix \( X := (\mathbb{R}^N, \| \cdot \|_X) \) and \( Y := (\mathbb{R}^M, \| \cdot \|_Y) \), where \( N, M \in \mathbb{N} \), and \( \| \cdot \|_X \) and \( \| \cdot \|_Y \) are some norms on \( \mathbb{R}^N \) and \( \mathbb{R}^M \), respectively. In the case of standard \( \ell^p \) spaces, with \( p \geq 1 \), then we will indicate the corresponding norm with the usual notation \( \| \cdot \|_p \).

We will indicate in bold any finite dimensional (column) vector, e.g. \( x := (x_1, \ldots, x_N)^T \in \mathbb{R}^N \), where \( T \) denotes the transpose operation, and we will use the notation \( x \leq \hat{x} \) meaning that \( x_i \leq \hat{x}_i \) for every \( i = 1, \ldots, N \). With abuse of language, given a real-valued function \( f : \mathbb{R}^N \to \mathbb{R} \),
\( \mathbb{R} \), we will say that \( \sigma \) is monotone nondecreasing if \( \sigma(x) \leq \sigma(\hat{x}) \) for every \( x \leq \hat{x} \). In case of a function with multivariate output, \( \sigma : \mathbb{R}^N \to \mathbb{R}^D \), we will indicate with \( \sigma_d : \mathbb{R}^N \to \mathbb{R}^d \) its components, for \( d = 1, \ldots, D \).

For a fixed \( z \in Z := (\mathbb{R}_D, \| \cdot \|_Z) \), we indicate with \( C(z, \cdot) : X \to \mathcal{X} \times \mathcal{X} \) the ‘concatenation’ operator, that is,

\[
C(z, x) := (z_1, \ldots, z_D, x_1, \ldots, x_N)^	op.
\]

Fix now a matrix \( F : X \to Y \), which is the discretization of an ill-posed linear operator between Banach spaces. We will assume that, given the unperturbed and observed data \( y \in Y \) and \( y^\delta \in Y \), respectively, then

\[
y \in \text{Rg}(F), \quad \text{that is,} \quad \text{Fx} = y \quad \text{is solvable,}
\]

and

\[
y^\delta = y + \eta \quad \text{where} \quad \|\eta\|_Y \leq \delta.
\]

We recall that a Banach space \( Y \) is uniformly smooth if its modulus of smoothness

\[
\rho(\tau) := \sup \left\{ \frac{\|y + \tau y\| + \|y - \tau y\|}{2} - 1 \mid \|y\| = \|\tau y\| = 1 \right\}, \quad \tau > 0
\]

satisfies \( \lim_{\tau \to 0^+} \rho(\tau) / \tau = 0 \). Examples of uniformly smooth spaces are all the \( \ell^p \)-spaces for \( p \in (1, \infty) \).

Given an extended real-valued function, \( \mathcal{R} : \text{dom}(\mathcal{R}) \subseteq X \to (-\infty, +\infty] \), then \( \mathcal{R} \) is uniformly convex if there exists a nonnegative map \( h : [0, +\infty) \to [0, +\infty] \) such that \( h(x) = 0 \) if and only if \( s = 0 \) and

\[
\mathcal{R}(x + (1 - t)\hat{x}) + t(1 - t)h(\|x - \hat{x}\|) \leq t\mathcal{R}(x) + (1 - t)\mathcal{R}(\hat{x}), \quad \forall t \in [0, 1] \quad \text{and} \quad \forall x, \hat{x} \in \text{dom}(\mathcal{R}).
\]

Finally, we recall that \( \mathcal{R} \) is coercive if it is bounded below on bounded sets and

\[
\lim \inf_{\|x\|_X \to +\infty} \frac{\mathcal{R}(x)}{\|x\|_X} = +\infty.
\]

### 2.1. Bregman distance

Given a convex function

\[
\mathcal{R} : \text{dom}(\mathcal{R}) \subseteq X \to (-\infty, +\infty],
\]

\( \mathcal{R} \) is called proper if \( \text{dom}(\mathcal{R}) := \{ x \in X : \mathcal{R}(x) < +\infty \} \neq \emptyset \). For every \( \hat{x} \in \text{dom}(\mathcal{R}) \), a subgradient of \( \mathcal{R} \) at \( \hat{x} \) is an element \( \xi \) of the dual space \( X^* \) such that

\[
\mathcal{R}(x) - \mathcal{R}(\hat{x}) - \langle \xi, x - \hat{x} \rangle \geq 0 \quad \forall x \in X,
\]

where the bracket is the evaluation of \( \xi \) at \( x - \hat{x} \). Clearly, since \( X \) is finite dimensional, then \( X \) is reflexive and \( \langle \cdot, \cdot \rangle \) is the standard inner product, that is,

\[
\langle \xi, x - \hat{x} \rangle = \sum_{i=1}^N \xi_i (x_i - \hat{x}_i).
\]

The collection of all subgradients of \( \mathcal{R} \) at \( \hat{x} \) is denoted by \( \partial \mathcal{R}(\hat{x}) \). The subdifferential of \( \mathcal{R} \) is the multi-valued map \( \partial \mathcal{R} : \text{dom}(\partial \mathcal{R}) \subseteq X \to 2^{X^*} \) such that
dom (\partial \mathcal{R}) := \{\hat{x} \in \text{dom}(\mathcal{R}) : \partial \mathcal{R}(\hat{x}) \neq \emptyset\}, \quad \hat{x} \mapsto \partial \mathcal{R}(\hat{x}).

Let us recall that if dom(\mathcal{R}) = X, then dom (\partial \mathcal{R}) = X (e.g. [50, Lemma 3.16]).

Finally, for every \(\hat{x} \in \text{dom}(\partial \mathcal{R})\) and \(\xi \in \partial \mathcal{R}(\hat{x})\), the Bregman distance \(B^R_\xi (\cdot, \hat{x}) : X \to [0, +\infty)\) induced by \(\mathcal{R}\) at \(\hat{x}\) in the direction \(\xi\) is defined by

\[B^R_\xi (x, \hat{x}) := \mathcal{R}(x) - \mathcal{R}(\hat{x}) - \langle \xi, x - \hat{x} \rangle.\]

**Remark 2.1.** It is straightforward to check that if \(\mathcal{R}\) is uniformly convex then \(B^R_\xi (\cdot, \hat{x})\) is uniformly convex too, for any fixed \(\xi\) and \(\hat{x}\). Moreover, since \(X\) is reflexive, then \(B^R_\xi (\cdot, \hat{x})\) is coercive. See for example [55, Corollary 2.4]

We can now introduce the definition of solution of the model problem (1.1), with respect to the Bregman distance from a reference initial guess.

**Definition 2.1.** Fix \(x_0 \in \text{dom}(\partial \mathcal{R})\), \(\xi_0 \in \partial \mathcal{R}(x_0)\). An element \(x^\dagger \in \text{dom}(\mathcal{R})\) is called a \(B^R_{\xi_0}\) minimizing solution of (1.1) if \(Fx^\dagger = y\) and

\[B^R_{\xi_0} (x^\dagger, x_0) = \min \left\{ B^R_{\xi_0} (x, x_0) : x \in \text{dom}(\mathcal{R}), Fx = y \right\}.\]

As a last piece of notation, we introduce the duality map. For every fixed \(r \in (1, \infty)\), the duality map \(J_r : X \to 2^X\) is given by

\[J_r (x) := \{\xi \in X^* \mid \|\xi\| = \|x\|_r^{-1} \text{ and } \langle \xi, x \rangle = \|x\|_r\}.\]

In particular, \(J_r\) is the subdifferential of the map \(x \mapsto \frac{\|x\|_r^r}{r}\).

**Remark 2.2.** If \(X\) is an \(\ell^p\) space, then \(J_r\) is single-valued and for \(r = 2\) it holds

\[J_2 (x) = \text{sgn}(x) \frac{|x|^{p-1}}{\|x\|_2^{p-2}}.\]

More generally, if a Banach space is uniformly smooth, then \(J_r\) is single-valued.

In view of the above remark and for the well-posedness of the iNETT method (see section 5), we will assume that

\(Y\) is uniformly smooth. (H1)

### 2.2. Neural networks

A neural network is a chain of compositions of affine operators and nonlinear operators. For an introduction to neural networks from an applied mathematical point of view, we refer to [31], whereas we refer to [8] for a focus on deep learning techniques for inverse problems. We present here the basic architecture upon which we will devise the neural networks to be implemented in iNETT. There are several many choices for the linear and the nonlinear operators, and each of them generate a different neural network. We do not focus now on those choices, which will be made only later (see sections 4, 5.3 and 6), to keep here a more general setting.

First, fix a set of parameters \(\Theta := \{b^k, A^{k, i}; W^k\}_{k=1}^L\), where \(b^k\) are vectors commonly called bias terms, and \(A^{k, i}\) and \(W^k\) are matrices. Second, fix a collection \(\{\sigma^k\}_{k=1}^L\) of possibly nonlinear operators.
Figure 1. Example of a simple neural network architecture of depth $L = 8$, with input $x$ and output $z$ described by the white and green rectangles, respectively. As it can be visually checked, the operators $A_{2,1}', A_{4,1}', A_{6,4}'$, and $C(z^3, z^7)$ bring over data of previous iterations skipping some connections. To differentiate the concatenation $C(\cdot, \cdot)$ from the other matrix-vector operators, we use a double edge to denote it. Inside the dotted square, it is depicted a blow-up of the second layer: The rows of the matrix $W_2 = (w_{2ij}) \in \mathbb{R}^{2 \times 3}$ are represented by the violet circles. Note that in that layer the bias term $\hat{b}_2$ is given by $\hat{b}_2 = b_2 + A_{2,1}'x$.

Finally, define $\Phi_\Theta : X \to Z$ such that

$$\begin{align*}
\Phi_\Theta(x) := z \in Z, \\
z^{k+1} = \sigma^k(\hat{b}^k + W^k z^k) \in Z^{k+1} & \quad \text{for } k = 1, \ldots, L, \\
z^i = C^i(z^i) := \begin{cases} 
z^i & \text{for } i \in \{1, \ldots, k\}, \\
C(z^i, \hat{z}^i) & \text{for } i \in \{1, \ldots, k\},
\end{cases} \\
\hat{b}^k = b^k + A_{k,j}^k z^j & \quad \text{for } j \in \{1, \ldots, k\}, \\
z^1 := x \in X,
\end{align*}$$

where $Z^{k+1} = Z := (\mathbb{R}^{D_k}, \| \cdot \|_Z)$ and $Z^k := (\mathbb{R}^{D_k}, \| \cdot \|_{Z^k})$ are finite dimensional normed vector spaces, and $C(\cdot, \cdot)$ is the concatenation operator (2.1). The operators $C(z^i, \cdot)$ and $A_{k,j}^k$ represent the skip connections, that is, some of the data in the previous iterations are used in future iterations, skipping intermediate steps. See figure 1 for a visual representation. The integer $L$
is referred to as the depth of the neural network, and \( \sigma^k (b^k + W^k z^k) \) as the kth layer. When \( L > 2 \), then the neural network is commonly called deep neural network.

The set \( \Theta \) is made by the disjoint union of two subsets, the set of free parameters \( \Theta_{\text{free}} \) and the set of frozen parameters \( \Theta_{\text{frozen}} \), that is

\[
\Theta = \Theta_{\text{free}} \sqcup \Theta_{\text{frozen}}.
\]

The set of free parameters \( \Theta_{\text{free}} \) is typically initialized to a starting set of values and then it is trained by minimizing a loss function over a training sample. On the other hand, the set of frozen parameters \( \Theta_{\text{frozen}} \) is fixed and unaffected by the training process. For example, some of the matrices \( W^k \) can be fixed to be the identity matrix \( I \) or the bias terms \( b^k \) to be the zero vector \( 0 \). \( \Theta_{\text{frozen}} \) can be empty, that is, all the parameters are trainable. About the specific training strategy we will employ, see section 5.3.

In the case that \( A^k \) is fixed to be the zero matrix and \( z^k = z^k \), for every \( k \), then we have a feedforward neural network. For examples of simple architectures of feedforward neural networks of convolutional type, see [19, 41]. For examples of more involved neural networks described by (NN), see ResNet [29], DenseNet [32] and U-Net [48].

### 3. Convex and uniformly convex neural networks

We provide some sufficient conditions to guarantee that a neural network, with the architecture described by (NN) in section 2.2, is component-wise convex or component-wise uniformly convex. In particular, given any neural network of depth \( L \), it is possible to generate a component-wise uniformly convex neural network of depth \( L + 1 \) which embeds the original neural network.

Fix a neural network of depth \( L \), \( \Phi_{\Theta}: X \to Z \), and consider the following constraints:

\( \sigma^k \) is continuous, convex and monotone nondecreasing \( \forall d = 1, \ldots, D_k, \forall k = 1, \ldots, L; \) \hspace{0.5cm} (C1)

\[
\begin{align*}
W^k & \text{ has nonnegative entries } \forall k = 2, \ldots, L, \\
A^k & \text{ has nonnegative entries } \forall k \geq 2.
\end{align*}
\] \hspace{0.5cm} (C2)

In the next proposition we show that \( \Phi_{\Theta} \) is component-wise convex. This result is a generalization of the input convex neural networks presented in [2], but we largely extend them to build modern architectures which can embrace state-of-the-art neural networks.

**Theorem 3.1.** Under the constraints (C1) and (C2), any neural network described by the set of equations (NN) is continuous and component-wise convex.

**Proof.** \( \Phi_{\Theta} \) is trivially continuous because composition of continuous functions. We want to show that given \( \Phi_{\Theta} = (\Phi_{\Theta,1}, \ldots, \Phi_{\Theta,D})^T \), then

\[
\Phi_{\Theta,d}(x + (1-t)\omega) \leq t\Phi_{\Theta,d}(x) + (1-t)\Phi_{\Theta,d}(\omega)
\]

for every \( d = 1, \ldots, D \) and \( x, \omega \in X \).

We proceed by induction. Fix \( L = 1 \) and \( z^1 := tx + (1-t)\omega \in X \). Then, by (NN), it holds that

\[
\Phi_{\Theta,d}(z^1) = \sigma^1_d \left( b^1 + W^1 z^1 \right) = \sigma^1_d \left( A^1 \cdot z^1 + W^1 C^1(z^1) \right).
\]

Let us remember that \( C^1 \) is such that

\[
\begin{align*}
C^1 &: X \to X, \\
C^1(x) &= x,
\end{align*}
\] or \[
\begin{align*}
C^1 &: X \to X \times X, \\
C^1(x) &= C(x, x).
\end{align*}
\]
In the latter case,
\[
C^1(z^1) = (tx + (1-t)\omega, tx + (1-t)\omega)^T
= t(x, x)^T + (1-t)(\omega, \omega)^T
= tC(x, x) + (1-t)C(\omega, \omega)
= tC^1(x) + (1-t)C^1(\omega).
\]

Then, by the linearity of \(A^{1,1}\) and \(W^1\), and independently of the choice of \(C^1\), it is clear that
\[
b^1 + A^{1,1}z^1 + W^1C^1(z^1) = t(b^1 + A^{1,1}x + W^1C^1(x)) + (1-t)(b^1 + A^{1,1}\omega + W^1C^1(\omega))
= mu + (1-t)v,
\]
where
\[
u := b^1 + A^{1,1}x + W^1C^1(x), \quad v := b^1 + A^{1,1}\omega + W^1C^1(\omega).
\]

Using now the convexity of \(\sigma_d^1\), we obtain
\[
\Phi_{\varepsilon_0,d}(tx + (1-t)\omega) = \sigma_d^1(b^1 + A^{1,1}z^1 + W^1C^1(z^1))
= \sigma_d^1(mu + (1-t)v)
\leq m_d^1(u) + (1-t)m_d^1(v)
= t\Phi_{\varepsilon_0,d}(x) + (1-t)\Phi_{\varepsilon_0,d}(\omega),
\]
namely, \(\Phi_{\varepsilon_0}\) is component-wise convex. Assume now that every neural network \(\Phi_{\varepsilon_0}\) described by (NN) is component-wise convex when the depth \(L\) is \(L = 1, \ldots, n-1\), for any fixed \(n \geq 2\). We need to prove that \(\Phi_{\varepsilon_0}\) is component-wise convex when \(L = n\). Fix again \(z^1 := tx + (1-t)\omega\). Observe that, at each step \(k = 1, \ldots, L - 1\), every \(z^{k+1}\) in (NN) is a neural network \(\Phi_{\varepsilon_0}^k : X \rightarrow Z^{k+1}\) of depth \(L_k \leq L - 1\) and with the same initial input \(z^1\), that is \(z^{k+1} = \Phi_{\varepsilon_0}^k(z^1)\). Then, by the induction hypothesis,
\[
\Phi_{\varepsilon_0}^k(z^1) = \Phi_{\varepsilon_0}^k(tx + (1-t)\omega) \leq t\Phi_{\varepsilon_0}^k(x) + (1-t)\Phi_{\varepsilon_0}^k(\omega) \quad \forall k = 1, \ldots, L - 1.
\]

So,
\[
\Phi_{\varepsilon_0}(z^1) = z^{L+1} = \sigma_d^L(b^L + W^Lz^L)
= \sigma_d^L(b^L + A^{L,1}z^L + W^LC^L(z^L)),
= \sigma_d^L(b^L + A^{L,1}\Phi_{\varepsilon_0}^{L-1}(z^1) + W^LC^L(\Phi_{\varepsilon_0}^{L-1}(z^1))).
\]

We need to take care of the concatenation operator. Again, there are two cases:
\[
\begin{cases}
C^L : Z^L \rightarrow Z^L; \\
C^L(\Phi_{\varepsilon_0}^{L-1}(z^1)) = \Phi_{\varepsilon_0}^{L-1}(z^1),
\end{cases}
\quad \text{or} \quad
\begin{cases}
C^L : Z^L \rightarrow Z^L \times Z^L; \\
C^L(\Phi_{\varepsilon_0}^{L-1}(z^1)) = C(\Phi_{\varepsilon_0}^{L-1}(z^1), \Phi_{\varepsilon_0}^{L-1}(z^1)),
\end{cases}
\]
where \(i_L \in \{1, \ldots, L\}\), and with the convention that \(\Phi_{\varepsilon_0}^0(z^1) = z^1\). Clearly,
\[
C(u_1, u_2) \leq C(v_1, v_2) \quad \text{for every} \quad u_1 \leq v_1, u_2 \leq v_2.
\]

Therefore, if we are in the latter case, by (3.1)
\[ C(\Phi_{\Theta}^{L-1}(z^l), \Phi_{\Theta}^{L-1}(z^l')) = C(\Phi_{\Theta}^{L-1}(ix + (1-t)\omega), \Phi_{\Theta}^{L-1}(ix + (1-t)\omega)) \]
\[ \leq C(r\Phi_{\Theta}^{L-1}(x) + (1-t)\Phi_{\Theta}^{L-1}(\omega), r\Phi_{\Theta}^{L-1}(x) + (1-t)\Phi_{\Theta}^{L-1}(\omega)) \]
\[ = rC(\Phi_{\Theta}^{L-1}(x), \Phi_{\Theta}^{L-1}(x)) + (1-r)C(\Phi_{\Theta}^{L-1}(\omega), \Phi_{\Theta}^{L-1}(\omega)) \]
\[ = rC(\Phi_{\Theta}^{L-1}(x)) + (1-r)C(\Phi_{\Theta}^{L-1}(\omega)). \]

As a consequence, independently from the case we are in,
\[ C^c(\Phi_{\Theta}^{L-1}(z^l)) \leq rC^c(\Phi_{\Theta}^{L-1}(x)) + (1-r)C^c(\Phi_{\Theta}^{L-1}(\omega)), \]
and then, by (C2),
\[ W_k C^c(\Phi_{\Theta}^{L-1}(z^l)) \leq W_k C^c(\Phi_{\Theta}^{L-1}(x)) + (1-t)W_k C^c(\Phi_{\Theta}^{L-1}(\omega)). \tag{3.2} \]

Now, instead, if \( j_L = 1 \), then \( \Phi_{\Theta}^{L-1}(z^l) = z^l \) and by linearity
\[ A_2^{L-1}\Phi_{\Theta}^{L-1}(z^l) = tA_2^{L-1}x + (1-t)A_2^{L-1}\omega. \tag{3.3} \]

If \( j_L \geq 2 \), by (3.1) and (C2),
\[ A_2^{L-1}\Phi_{\Theta}^{L-1}(z^l) \leq tA_2^{L-1}\Phi_{\Theta}^{L-1}(x) + (1-t)A_2^{L-1}\Phi_{\Theta}^{L-1}(\omega). \tag{3.4} \]

Let us write
\[ u := b^L + A_2^{L-1}\Phi_{\Theta}^{L-1}(x) + W_k C^c(\Phi_{\Theta}^{L-1}(x)), \]
\[ v := b^L + A_2^{L-1}\Phi_{\Theta}^{L-1}(\omega) + W_k C^c(\Phi_{\Theta}^{L-1}(\omega)). \]

Combining (3.2), (3.3) and (3.4) with (C1), we get
\[ \Phi_{\Theta,\omega}(z^l) = \sigma^L_0(\Phi^{L-1}(z^l)) \leq \sigma^L_0(u + (1-t)v) \]
\[ \leq r\sigma^L_0(u) + (1-r)\sigma^L_0(v) \]
\[ = r\Phi_{\Theta,\omega}(x) + (1-r)\Phi_{\Theta,\omega}(\omega). \]

This concludes the proof. \qed

See figure 2 for a visual representation of a component-wise convex neural network. Now, given a component-wise convex neural network of depth \( L \), define
\[ \begin{align*}
\Phi_{\Theta}^L : X &\rightarrow Z', \\
\Phi_{\Theta}^L(x) &:= \Sigma_{f_{Z'}}(x, \Phi_{\Theta}^L(x)),
\end{align*} \tag{ucNN} \]
where \( Z' := (\mathbb{R}^{D'}, \| \cdot \|_{Z'}) \) and \( \Sigma_{f_{Z'}} : X \times Z \rightarrow Z' \) is given by
\[ \Sigma_{f_{Z'}}(x, z) := (f_1(x) + g_1(z), \ldots, f_{D'}(x) + g_{D'}(z))^T, \]
and \( f_d : X \rightarrow \mathbb{R}, g_d : Z \rightarrow \mathbb{R} \) are generic real-valued functions for \( d = 1, \ldots, D' \). In particular, \( \Phi_{\Theta}^L \) is a neural network of depth \( L + 1 \), whose last layer \( \sigma^{L+1}(b^{L+1} + W^{L+1}z^{L+1}) \) is given by
\[ \begin{align*}
\begin{cases}
\sigma^{L+1} = C(z^l, z^{L+1}), \\
W^{L+1} = I, \\
b^{L+1} = A^{L+1, j_L+1} = 0, \\
\end{cases}
\end{align*} \]
All the functions $\sigma_k$ are now assumed to be component-wise convex and monotone nondecreasing, and this is represented by the triangular-shaped nodes. Observe that the matrices $W^k$, for $k = 2, \ldots, 8$, and $A^6$, have nonnegative entries, while $W^1$, $A^2$, $A^4$, and the bias terms $b^k$ do not have any constraints.

Consider the following assumptions, for every $d = 1, \ldots, D'$:

- $f_d$ continuous and uniformly convex; \hspace{1cm} (C3)
- $g_d$ continuous, convex and monotone nondecreasing on the image $\Phi_\Theta(X) \subseteq Z$. \hspace{1cm} (C4)

**Theorem 3.2.** Under the constraints (C3) and (C4), the neural network $\Phi_\Theta^{uc}$ defined in (ucNN) is component-wise uniformly convex.

**Proof.** Given a component-wise convex map $\Phi_\Theta : X \to Z$, by direct computation it is immediate to check that $f_d + g_d \circ \Phi_\Theta$ is uniformly convex for every $d = 1, \ldots, D'$, under the constraints (C3) and (C4).

Combining theorems 3.1 and 3.2, given a neural network $\Phi_\Theta : X \to Z$, it is now easy to build a continuous and component-wise uniformly convex neural network $\Phi_\Theta^{uc} : X \to Z'$ by employing the following steps:

1. Fix a neural network $\Phi_\Theta : X \to Z$ whose architecture is given by (NN).
2. Impose the constraints (C1)–(C2) to get $\Phi_\Theta^{uc} : X \to Z$. 

Figure 2. Continuation of the example in figure 1. All the functions $\sigma^k$ are now assumed to be component-wise convex and monotone nondecreasing, and this is represented by the triangular-shaped nodes. Observe that the matrices $W^k$, for $k = 2, \ldots, 8$, and $A^6, A^4, A^2, A^4, A^4, A^2$ have nonnegative entries, while $W^1, A^2, A^4, A^4, A^2$ and the bias terms $b^k$ do not have any constraints.
Define $\Phi^{uc}: X \rightarrow Z^{'}$ by equation (ucNN) and impose the constraints (C3)–(C4).

See figure 3 for a visual representation of a component-wise uniformly convex neural network.

4. Example: Convex and uniformly convex U-net with batch normalization (BN)

Here we provide particular examples of convex and uniformly convex U-net, which we will use in sections 5.3 and 6. U-net is a convolutional neural network originally designed for biomedical image segmentation [48]. The network consists of a contracting path and a symmetric expanding path so that it has a U-shaped architecture. Several variants of the U-net architecture have been efficiently implemented in the solutions of inverse problems, e.g. [5, 7, 30, 44, 54].
We propose a modified U-net architecture as shown in figure 4, and we will explain the approaches to obtain convex and uniformly convex U-net architectures according to the general theories proposed in section 3. In section 4.1, we give a brief description of the BN which is one of the extra features we add to the modified U-net. In section 4.2, we provide a mathematical modeling of the modified U-net architecture using the notations of general neural networks introduced in section 2.2. Then in sections 4.3 and 4.4, we explain the approaches to achieve convex and uniformly convex U-net architectures.

4.1. BN

BN is a state-of-the-art technique for accelerating the training speed and improving the performance of optimization [33]. The BN works differently in the training process and in the inference process after training. Hereby we give a brief description of the BN operation according to our notations for general neural networks in equation (NN).

With a given batch size $n_0$ and a mini-batch of the initial input, $B = \{x_1, \ldots, x_{n_0}\}$, consider the output after convolution at the $k$th layer of the neural network $\Phi_\Theta$,

$$B^k = \{u^k_1, \ldots, u^k_{n_0}\}, \text{ where } u^k_j := b^k + W^k z^k_j, \quad j = 1, \ldots, n_0.$$

The operation of BN reads as follows,

$$BN(u) := \beta^k + \text{diag}(\gamma^k)\frac{u - E(B^k)}{\sqrt{\text{Var}(B^k) + \epsilon}}, \quad \forall u \in B^k,$$

where $\gamma^k$ and $\beta^k$ are column vectors and the free parameters to be learned, $E(B^k)$ and $\text{Var}(B^k)$ denote mean value and variance of the elements in $B^k$, respectively, and $\epsilon > 0$ is a small constant introduced for numerical stability.

In the training process, since $E(B^k)$ and $\text{Var}(B^k)$ will involve the variable $u \in B^k$, BN is not necessarily component-wise convex nor monotone nondecreasing. After the training process
is completed, \( E(B^t) \) and \( \text{Var}(B^t) \) will be fixed using the full population rather than the mini-batch, and so BN reduces to an affine transformation.

4.2. The modified U-net architecture

We provide a mathematical modeling of the modified U-net architecture \( \Phi_{\Theta} \) as shown in figure 4. Here we will make use of the notations of general neural networks introduced in section 2.2. Before proceeding further, it is necessary to briefly introduce the following operators.

- **Convolution:** It is a linear operator which can be represented by a tensor \( K \) of dimension \( f_1 \times f_2 \times c_1 \times c_2 \), where \( f_1 \times f_2 \) is the filter size of the convolution, \( c_1 \) and \( c_2 \) denote the number of channels of the input and output images, respectively. More details about the convolution operator can be found in [31, section 7] and [21]. In the U-net architecture, a bias term is added after every convolution.

- **Zero padding:** It is a linear operator which can be achieved by embedding a given matrix of dimension \( n \times m \) into a zero matrix of dimension \( n' \times m' \), with \( n' \geq n \) and \( m' \geq m \). By reshaping the matrices before and after zero padding to be column vectors, zero padding can be expressed as a matrix product, \( \mathbf{M} = \mathbf{P}_0 \mathbf{N} \), where \( \mathbf{P}_0 \) is a binary matrix with entries from \( \{0,1\} \).

- **Rectified linear unit (ReLU):** \( x \mapsto \max\{0,x\} \), where the maximum is applied component-wise.

- **max-pool:** Given a matrix of real numbers, partition it into \( i \times j \) sub-matrices; the \( \text{max-pool}_{i \times j}(\cdot) \) returns a matrix that calculates the maximum value for each sub-matrix in the \( i \times j \) partition.

- **BN:** Details are given in section 4.1. We emphasis that BN reduces to an affine transformation after training.

In the modified U-net architecture \( \Phi_{\Theta} \), every convolution operator is combined with a zero padding, such that the input and output images have the same size. In addition, we add a BN between each convolution and ReLU function. As shown in figure 4, the modified U-net architecture can be constructed by several unit blocks: right arrow (involving concatenation), down arrow, and up arrow. Accordingly, we provide mathematical modeling for these unit blocks.

- **Right arrow \( \rightarrow \):** A \( \ell \)th right arrow reads as

\[
\text{ReLU}(\text{BN}(b^k + K^k \hat{z}^k)).
\]

According to the notations in equation (NN), the block of right arrow can be modeled as the composition of two layers,

\[
\begin{align*}
\hat{z}^{k+1} &= \sigma^k (b^k + W^k \hat{z}^k), \\
\hat{z}^{k+1} &= \hat{z}^{k+1} \\
\hat{z}^{k+2} &= \sigma^{k+1} (b^{k+1} + W^{k+1} \hat{z}^{k+1})
\end{align*}
\] (4.1)
where

\[
\begin{align*}
W^k &= K^k P_0, \quad \hat{b}^k = b^k, \\
\sigma^k(u) &= \frac{u - E(B^k)}{\sqrt{Var(B^k)} + \epsilon}, \quad \text{and} \\
K^k, b^k &\in \Theta_{\text{free}}, \quad P_0 \in \Theta_{\text{frozen}} \\
W^{k+1} &= \text{diag}(\gamma^k), \quad \hat{b}^{k+1} = \beta^k
\end{align*}
\]

\(P_0\) denotes the matrix of zero padding, which is a binary matrix with entries from \(\{0, 1\}\). The filter size of every convolution \(K^k\) is \(3 \times 3\) and the stride is 1. In the ascending part of the U-net architecture, the concatenation operation is involved before the first convolution of each layer, and it can be modeled by the formulation of general neural networks as shown in section 2.2, i.e. \(\hat{z}^k = C(\hat{z}^k, \hat{z}^k)\).

- Down arrow \(\downarrow\): An \(i\)th down arrow reads as

\[
\text{max-pool}_{2 \times 2}(\hat{z}^i).
\]

It can be simply modeled by the formula of general neural networks as shown in equation (NN),

\[
\sigma^i(\hat{b}^i + W^i \hat{z}^i), \tag{4.2}
\]

where \(\hat{b}^i = 0, \quad W^i = I, \quad \text{and} \quad \sigma^i(u) = \text{max-pool}_{2 \times 2}(u); \sigma^i\) is continuous, component-wise convex and monotone nondecreasing.

- Up arrow \(\uparrow\): The up arrow has the same structure of the right arrow, and it can be modeled in the same way by equation (4.1). There are some minor differences in the convolution \(K^k\):

(i) the filter size of \(K^k\) is \(2 \times 2\); (ii) the way of zero padding is different from that in the right arrow, but since it is still a zero padding, the matrix \(P_0\) is still a binary matrix with entries from \(\{0, 1\}\).

- Last right arrow \(\Longrightarrow\): The last right arrow is just a convolution with filter size \(1 \times 1\). It can be modeled as part of equation (4.1):

\[
z^{L+1} = \sigma^L(\hat{b}^L + W^L z^L), \tag{4.3}
\]

where \(W^L = K^L P_0, \quad \hat{b}^L = b^L, \quad \text{and} \quad \sigma^L(u) = u\). The output \(z^{L+1}\) is of size \(256 \times 256 \times 1\), where the channel number is 1. As a result, the bias term \(\hat{b}^L = b^L\) is essentially a scalar, i.e. the components of \(b^L\) are all equal to \(b_0^L \in \mathbb{R}\).

### 4.3. Convex U-net

In section 4.2, we showed that the modified U-net architecture \(\Phi_{\Theta}\) can be modeled by the set of equations (NN). Then according to theorem 3.1, we can achieve a convex U-net by imposing the constraints (C1) and (C2). Since the skip connection described by \(A^{k,h}\) is not involved in \(\Phi_{\Theta}\), we only need to consider the properties of \(\sigma^k\) and \(W^k\).

The architecture modeled by equation (4.2) automatically satisfies the constraints (C1) and (C2). For the architecture modeled by equation (4.1) (which includes the formulation in equation (4.3) as well), \(\sigma^k\) and \(\sigma^{k+1}\) are continuous, convex, and monotone nondecreasing after training, so that the constraint (C1) is satisfied. To impose the constraint (C2) on \(W^k\) and \(W^{k+1}\), we have to impose nonnegativity constraint on the entries of \(K^k\) and \(\gamma^k\). Above all, we have the following conclusion.
Proposition 4.1. By imposing nonnegativity constraint on \( \{ K^k \mid \forall k \geq 2 \} \) and \( \{ \gamma^k \mid \forall k \geq 1 \} \), the modified U-net \( \Phi_{\Theta} \) turns out to be a convex U-net in the inference process after training.

In next subsection, we will build a uniformly convex U-net by employing the architecture of convex U-net. We will ask the output of the convex U-net to be nonnegative, which is helpful to construct the uniformly convex U-net in a very simple way. Considering the convex U-net obtained in proposition 4.1, thanks to the ReLU function applied just before the last convolution operator, and then the nonnegativity of the entries of the last convolution operator itself, the output \( z^{l+1} \) must be nonnegative if we further impose nonnegativity constraint on the last bias term \( b^l \).

Proposition 4.2. By imposing nonnegativity constraint on \( \{ K^k \mid \forall k \geq 2 \} \), \( \{ \gamma^k \mid \forall k \geq 1 \} \) and \( b^l \), the modified U-net \( \Phi_{\Theta} \) turns out to be a convex U-net \( \Phi_{\Theta}^c \) having nonnegative output, i.e. the components of \( z^{l+1} = \Phi_{\Theta}^c(x) \) are all nonnegative.

4.4. Uniformly convex U-net

With the trained convex U-net \( \Phi_{\Theta}^c \) having nonnegative output, one can construct a uniformly convex U-net according to the following formula,

\[
\Phi_{\Theta}^{uc} : X \to (\mathbb{R}, | \cdot |),
\]

where \( D' = 1 \), \( f_1(x) = a\|x\|^p \), and \( g_1(z) = \|z\|^q \).

The function \( f_1(x) = a\|x\|^p \) is uniformly convex for \( p \geq 2 \); e.g. see [15, Theorem 2.3] and [39, Theorem 1.f.1]. The function \( g_1(z) = \|z\|^q \) is continuous, convex and monotone nondecreasing in the domain \( \mathbb{R}^N_0 := \{ z \in \mathbb{R}^N \mid z_i \geq 0, i = 1, \ldots, N \} \). Moreover, as shown in proposition 4.2, the output of the convex U-net \( \Phi_{\Theta}^c \) satisfies \( \Phi_{\Theta}^c \in \mathbb{R}^N_0 \). As a result, \( \Phi_{\Theta}^{uc} \) in equation (4.4) is uniformly convex according to theorem 3.2. Equation (4.4) gives a very simple way to construct the uniformly convex U-net.

5. iNETT method

In this section we propose the iNETT method. First, fix a continuous, uniformly convex and well trained neural network

\[
\Phi_{\Theta}^{uc} : X \to (\mathbb{R}, | \cdot |),
\]

for example by applying the steps (s1), (s2) and (s3) at the end of section 3, and define

\[
\mathcal{R} := \Phi_{\Theta}^{uc}.
\]

For the numerical experiments in section 6, we will adopt the modified U-net architecture presented in section 4 as equation (4.4).

Fix \( r \in (1, \infty) \), \( x_0 \in X \), \( \zeta_0 \in \partial \mathcal{R}(x_0) \) and a sequence of positive real numbers \( \{ \alpha_n \} \) such that

\[
\sum_{n=1}^{\infty} \alpha_n^{-1} = \infty \quad \text{and} \quad \alpha_n \leq c \alpha_{n+1}, \quad \text{where} \ c > 0 \ \text{is a constant}.
\]
The iNETT method is then given by

\[
\begin{align*}
x^\delta_n &= \arg\min_{x \in X} \frac{1}{2} \|Fx - y^\delta\|_F^2 + \alpha_n B^R_{\xi_{n-1}}(x, x^\delta_{n-1}), \\
\xi^\delta_n &= \xi^\delta_{n-1} - \frac{1}{\alpha_n} F^T J_t (Fx^\delta_n - y^\delta),
\end{align*}
\]  

(iNETT)

The stopping rule associated to iNETT is the standard discrepancy principle. Specifically, we stop the iterations at the first step \(n_\delta = n\) such that

\[ \|Fx^\delta_n - y^\delta\|_Y \leq \tau \delta < \|Fx^\delta_{n-1} - y^\delta\|_Y, \tag{5.1} \]

where \(\tau > 1\) is a fixed constant, and \(\delta\) is the parameter which estimates the level of noise \(\eta\), viz. \(\|\eta\|_Y = \|y^\delta - y\|_Y \leq \delta\). Following we will provide convergence analysis and implementation details of the iNETT method.

### 5.1. Convergence analysis

We present the proof of well-posedness, convergence and stability of the iNETT method. The proof is mainly a straightforward adaptation, in this finite dimensional setting, of standard results of general convex regularization theory (e.g. [50] and [35, Theorem 3.2]). For the convenience of the reader and to make the theoretical treatment self-contained, we develop here the main points.

**Theorem 5.1.** Let \(F : X \to Y\) be a linear operator between finite dimensional normed spaces and suppose that (H0) and (H1) are valid. Let \( := \Phi\) be a continuous and uniformly convex neural network. Then the method iNETT is well-posed and there exists a unique \(B^R_{\xi_n}\)-minimizing solution \(x^1\). The method iNETT, coupled with the stopping rule (5.1), stops in finite steps \(n_\delta = n\) and it converges to \(x^1\) as \(\delta \to 0\). In particular,

\[ x^\delta_{n_\delta} \to x^1, \quad \mathcal{R}(x^\delta_{n_\delta}) \to \mathcal{R}(x^1), \quad B^R_{\xi_{n_\delta}}(x^1, x^\delta_{n_\delta}) \to 0 \]

as \(\delta \to 0\).

**Proof.** About the existence of \(x^1\) in definition 2.1, observe that

\[ 0 \leq c := \inf \left\{ B^R_{\xi_0}(x, x_0) \mid x \in X, Fx = y \right\} \]

is well-defined because of (H0). Since \(B^R_{\xi_0}(x, x_0)\) is continuous and coercive (see remark 2.1), the level sets \(\{x \mid B^R_{\xi_0}(x, x_0) \leq M\}\) are compact for any \(M > 0\), and therefore there exists a minimizer \(x^1\), by standard topological arguments, which is unique because of the linearity of \(F\) and the uniform (therefore strict) convexity of \(\mathcal{R}\). In the same fashion, and because of the convexity of \(B^R_{\xi_n}\), \(x^\delta_n\) is well-defined and unique at each step.

Observe now that \(J_t\) is a single-valued map thanks to (H1), and that \(\xi^\delta_n \in \partial \mathcal{R}(x^\delta_n)\) in virtue of the minimality of \(x^\delta_n\). Indeed, if we define

\[
\mathcal{G}(x) := \frac{1}{2} \|Fx - y^\delta\|_F^2 + \alpha_n B^R_{\xi_{n-1}}(x, x^\delta_{n-1}),
\]

by subgradient calculus (see for example [47, Theorem 10.6]), and because \(J_t\) is the subgradient of the map \(y \mapsto \frac{1}{2} \|y\|_F^2\), it holds that

\[ 0 \in \partial \mathcal{G}(x^\delta_n) = F^T J_t (Fx^\delta_n - y^\delta) + \alpha_n \partial \mathcal{R}(x^\delta_n) - \alpha_n \xi^\delta_{n-1}. \]

The rest of the proof is a straightforward application of [35, Theorem 3.2].

5.2. Implementation

Since our applications in section 6 will be based on computerized tomography, from now on we will consider X and Y as vector spaces of digital images. We will use the standard $\ell^2$-norm for X, that is, $X = (\mathbb{R}^N, \| \cdot \|_2)$ and a normalized $\ell^2$-norm for $Y = (\mathbb{R}^M, \| \cdot \|_2)$ as $\| y \|_Y = \| y \|_2 / \sqrt{M}$, and we will fix $r = 2$. Therefore, the iNETT iteration will assume the following form (see also Remark 2.2),

$$
\begin{align*}
\begin{cases}
x_n^\delta = \arg \min_{x \in X} \| Fx - y^\delta \|_2^2 + \alpha_n B^R_{\xi_{n-1}} (x, x_{n-1}^\delta), \\
\xi_n^\delta = \xi_{n-1}^\delta - \frac{1}{\alpha_n} F^T (Fx_n^\delta - y^\delta),
\end{cases}
\end{align*}
$$

(iNETT)

The minimum problem in iNETT can be solved by a gradient descent approach. Recalling that the Bregman distance of the regularization term is defined as

$$B^R_{\xi_{n-1}} (x, x_{n-1}^\delta) = \mathcal{R}(x) - \mathcal{R}(x_{n-1}^\delta) - \langle \xi_{n-1}, x - x_{n-1}^\delta \rangle,$$

the gradient descent algorithm for $x_n^\delta$ has the following form,

$$x_{n,k+1}^\delta = x_{n,k}^\delta - s \left[ 1 M F^T (Fx_{n,k}^\delta - y^\delta) + \alpha_n \left( \frac{\partial \mathcal{R}(x)}{\partial x} \bigg|_{x = x_{n,k}^\delta} - \xi_{n-1}^\delta \right) \right], \quad k \in \mathbb{N}
$$

where $s > 0$ denotes the step size of the gradient descent iteration. The initial guess for the iteration can be taken as $x_{n,0}^\delta = x_{n-1}^\delta$.

5.3. Building a uniformly convex neural network regularizer

We will adopt the uniformly convex neural network (4.4) presented in section 4.4, with $p = q = 2$. That is,

$$\mathcal{R}(x) := \Phi_0^\{a\}(x) = a \| x \|_2^2 + \| \Phi_0^\{a\}(x) \|_2^2,$$

where $\Phi_0^\{a\}(x)$ is a well trained convex U-net having nonnegative output as shown in Proposition 4.2, and $a > 0$ is a small positive constant.

Since the regularizer is designed to penalize artifacts in the solution, the trained U-net $\Phi_0^\{a\}(x)$ should have small output for artifact-free images and have large output for images with artifacts. We follow the strategy proposed in [37] for the preparation of training data. Let $\{ x_s^\ast \ s = 1, \ldots, N_1 + N_2 \}$ denote a set of images in X. The training images are constructed in the following way,

$$z_s = \begin{cases} 
F^\dagger (Fx_s^\ast + \eta), & s = 1, \ldots, N_1 \\
x_s^\ast, & s = N_1 + 1, \ldots, N_1 + N_2
\end{cases},
$$

(5.2)

where $\eta$ denotes noise perturbations on the data, and $F^\dagger$ denotes the pseudo inverse of the forward operator $F$. In the set of training images $\{ z_s \ s = 1, \ldots, N_1 + N_2 \}$, the first $N_1$ $z_s$’s simulate the images with artifacts, while the last $N_2$ $z_s$’s simulate the artifact-free images. The output label is the error between $z_s$ and $x_s^\ast$,

$$r_s = \| x_s^\ast - z_s \|, \quad s = 1, \ldots, N_1 + N_2,
$$

(5.3)

where the absolute value is applied component-wise, and the training set is composed of the pairs of input images and output labels:

$$\{(z_s, r_s) \ s = 1, \ldots, N_1 + N_2\}.$$
Algorithm 1. Training the convex U-net $\Phi_{\Theta_{\text{free}}}^t$ having nonnegative output.

1: Initialize the free parameters set $\Theta_{\text{free}} = \{\beta^k; \gamma^k; b^k; K^k\}_k$.
2: for epoch $e = 1, 2, \ldots, N_e$ do
3: Sample a random permutation $\{\pi(1), \pi(2), \ldots, \pi(N_1 + N_2)\}$ of the index set $\{1, 2, \ldots, N_1 + N_2\}$.
4: With a given batch size $n_0$, partition the shuffled index set into $n = \frac{N_1 + N_2}{n_0}$ subsets: $\mathcal{I}_t = \{\pi((t-1)n_0 + 1), \ldots, \pi(tn_0)\}, t = 1, 2, \ldots, n$.
5: for $t = 1$ to $n$ do
6: Update $\Theta_{\text{free}}$ according to the Adam algorithm and mini-batch cost function $L_t(\Theta)$.
7: Impose nonnegativity constraint: $K^k = \max\{0, K^k\}, \forall k \geq 2$; $\gamma^k = \max\{0, \gamma^k\}, \forall k > 1$; $b^t = \max\{0, b^t\}$, where $b^t$ denotes the bias term of the last layer.
8: end for
9: end for
10: Fix $E(B^t)$ and $\text{Var}(B^t)$ in batch normalization (BN).

The loss function of one training sample is defined as follows,

$$L_t(\Theta) = \frac{1}{N} \| \Phi_{\Theta_{\text{free}}}^t(z_s) - r_s \|^2_2 + \lambda \| \tilde{\Theta} \|^2_2,$$

where

$$\tilde{\Theta} = \{K^k\}_k \subset \Theta_{\text{free}} = \{\beta^k; \gamma^k; b^k; K^k\}_k$$

considering the modified U-net architecture as shown in section 4, and $N$ denotes the pixel number of an image. We propose to use a mini-batch optimization approach in the training process, so that the cost function for updating $\Theta_{\text{free}}$ has the following form,

$$L_t(\Theta) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} L_t(\Theta) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \frac{1}{N} \| \Phi_{\Theta_{\text{free}}}^t(z_s) - r_s \|^2_2 + \lambda \| \tilde{\Theta} \|^2_2,$$  (5.4)

where $\mathcal{I}_t \subset \{1, 2, \ldots, N_1 + N_2\}$ denotes a mini-batch of the index set, and $|\mathcal{I}_t|$ denotes the batch size. The parameter $\lambda$ controls the amount of regularization applied. We utilize a random shuffling strategy [38, 49] for the sampling of the mini-batch set $\mathcal{I}_t$. Then the set of training parameters $\Theta_{\text{free}}$ is updated according to the Adam algorithm [36]. After each updating, we impose non-negativity constraint on $\gamma^k$ and $K^k$ for the convexity of the U-net $\Phi_{\Theta_{\text{free}}}^t$, with the only exception of the convolution matrix $K^1$ in the first layer. In addition, to make the convex U-net $\Phi_{\Theta_{\text{free}}}^t$ have nonnegative output, we impose non-negativity constraint on the bias term of the last layer $b^t$. After the training process is completed, we fix the elements $E(B^t)$ and $\text{Var}(B^t)$ which define the BN operators. Algorithm 1 summarizes the details of the training process.

5.4. Some remarks

As final comments for this section, we provide discussions on two subjects: (1) similarities and differences between NETT and iNETT; (2) alternative training strategies for the iNETT method.

5.4.1. NETT and iNETT. The NETT method in [37] is developed in the setting of infinite dimensional Banach spaces. The iNETT method can be generalized to the infinite dimensional
case as well, where one may require the Banach space $X$ to be reflexive, and re-define the neural network architecture (NN) in a proper way. In this paper, we propose iNETT in a finite dimensional case mainly for personal taste. Since we focus on the inverse problems in imaging science, e.g. computerized tomography, the setup is naturally of finite dimension. The images are defined in a finite dimensional vector space, and the neural networks are trained and deployed in the set of images with fixed finite pixels. When considering the accuracy of discretization, it will be necessary to study iNETT in the infinite dimensional setup. A main difficulty will be the convergence analysis: It is unclear under what hypotheses a convex neural network, coupled with training and optimization rules, converges to an infinite dimensional operator.

A major difference between NETT and iNETT lies in the assumption of coercivity or convexity on the data-driven regularizer. For the convergence analysis of NETT method, the regularization term $R(x) := \psi(\Phi_\Theta(x))$ is assumed to be coercive [37]. On the other hand, in our iNETT method, the neural network regularizer $R(x) := \Phi_{uc_\Theta}(x)$ is required to be uniformly convex. In fact, at each iteration of iNETT, the penalty term $P_n(x) := B^{R}_{\xi_{n-1}}(x, x_{n-1})$ is coercive, as a consequence of the uniform convexity of $R$ (see remark 2.1). The assumption of convexity on $\mathcal{R}$ is from the iterated Tikhonov method. For example, considering the proofs of [35, Theorems 3.1 and 3.2], a crucial role to prove the (strong) convergence is played by the following inequality,

$$B^{R}_{\xi}(x, \hat{x}) \geq f(||x - \hat{x}||),$$

where $f: [0, +\infty) \to [0, +\infty)$ is a strictly increasing continuous function such that $f(0) = 0$. The above inequality is true if and only if $R$ is uniformly convex (see [56, Theorem 3.5.10]). It would be interesting to understand if the assumption of uniform convexity on $\mathcal{R}$ can be relaxed, say for example just asking coercivity. To the best of our knowledge, it is yet an open problem.

5.4.2. Alternative training strategies for iNETT. As described in section 5.3, the training strategy of iNETT is adapted from that of NETT [37]. This is a natural choice when we consider the evolution of NETT into an iterated Tikhonov method. In this way, we can have a simple comparison between the reconstruction results of NETT and iNETT, which will be provided in the next section.

Nevertheless, it would be useful and interesting to study alternative training strategies for iNETT. In [40], it proposes an adversarial training strategy for the neural network regularizer. The adversarial regularizer learns to discriminate between the distribution of ground truth images and the distribution of unregularized reconstructions. The training can be performed even if only unsupervised training data are available [40]. In [42], the adversarial strategy is adopted to successfully build a learned convex regularizer. This strategy deserves elaborate study in the future work, and it provides a direction to further improve the iNETT algorithm.

Moreover, in [52], it develops a deep ADMM-Net for compressive sensing MRI, which is one of the earliest learned reconstruction methods in imaging science. The ADMM-net maps the ADMM iterative procedure to a data flow graph, and then generalizes the operations of ADMM to have learnable parameters as network layers. This strategy inspires us to generalize iNETT to a fully learned architecture in the future study, e.g. learn the neural network regularizer $R(x)$ in the optimization process, which would provide more flexibility of building the data-driven regularizer, even though the convergence analysis can become much more difficult.
6. Application to computerized tomography

We provide applications of the iNETT algorithm, as presented in section 5.2, in 2D computerized tomography (CT). The imaging problem can be modeled as an inverse problem of the Radon transform:

\[ y(s, \phi) = \int_{\mathbb{R}} x(s \cos \phi - t \sin \phi, s \sin \phi + t \cos \phi) \, dt, \quad (s, \phi) \in [0, 2\pi], \]  

(6.1)

where \( x(\cdot, \cdot) \) denotes the attenuation image to be recovered, and \( y \) denotes the projection data. After discretization, equation (6.1) reduces to a linear system,

\[ y = Fx, \]  

(6.2)

where \( x = (x_1, \ldots, x_N)^T \) is the vector of the attenuation image, \( y = (y_1, \ldots, y_M)^T \) simulates the measurement projection data, and \( F = (a_{ij})_{M \times N} \) denotes the projection matrix; the component \( a_{ij} \) represents the contribution of the \( j \)-th pixel of the attenuation image to the \( i \)-th projection datum. The inverse problem is to recover \( x \) from the measurement data \( y_\delta \) perturbed by unknown noises, viz. \( y_\delta = y + \eta \). In our CT application, we consider the attenuation image with \( 256 \times 256 \) pixels, and the sinogram of projection data is of size \( 256 \times 60 \) pixels. As a result, we have \( N = 256^2, M = 256 \times 60 \).

6.1. Training convex U-net

The convex U-net for iNETT is trained according to the approach described in section 5.3. The attenuation images are taken as synthetic phantoms consisting of randomly generated piecewise constant ellipses. We prepared \( 2600 \) synthetic phantoms, denoted as \( f_{x^*_s} \), and figure 5 shows six of them.

The pairs of input images and output labels \( \{ (z_s, r_s) \mid s = 1, \ldots, 2600 \} \) are constructed according to equations (5.2) and (5.3), where we take \( N_1 = N_2 = 1300 \), so that the first 1300 samples simulate the images with artifacts and the second 1300 samples simulate the artifact-free images. The input images of the first 1300 samples are constructed according to

\[ z_s = F^\dagger(Fx^*_s + \eta), \]

where we add 0%–10% Gaussian noise to simulate the real situation of practical measurements. To be specific,

\[ \eta = \eta \cdot (Fx^*_s) \cdot \mathcal{N}(0, 1), \]

where \( \eta \) denotes a random number between 0 and 0.1, and \( \mathcal{N}(0, 1) \) denotes the Gaussian noise with zero mean and unit variance. Figure 6(a) shows the simulated measurement data with Gaussian noises for the six synthetic phantoms displayed in figure 5. The pseudo inverse \( F^\dagger \) for the radon transform is achieved by the algebraic reconstruction technique (ART) [7, 25].
Figure 6. Measurement data and input images with artifacts corresponding to the six synthetic phantoms displayed in figure 5. (a) Measurement data with Gaussian noises: $Fx^* + \eta$; (b) input images with artifacts constructed by 5 rounds of ART iterations on the measurement data.

which is a Kaczmarz iterative scheme for the linear system (6.2). The ART algorithm has the following formula:

$$x^{(n)} = x^{(n-1)} + \frac{y^i - a_i^T x^{(n-1)}}{|a_i|^2} a_i, \quad n \in \mathbb{N}^+, \ i = 1, 2, \ldots, M,$$

where $a_i$ denotes the $i$th row of the projection matrix $F$, $a_i = (a_{i1}, \ldots, a_{iN})^T$, and $y^i$ denotes the $i$th component of the measurement data. One round of ART iterations implies one complete sweep of the measurement data $y^i$, with $i$ going from 1 to $M$. In our application, the pseudo inverse $F^\dagger$ is constructed by 5 rounds of ART iterations, and the initial guess is taken as $x^{(0)} = \frac{1}{N}1_{N\times 1}$. Figure 6(b) shows the input images $z_i = F^\dagger (Fx^* + \eta)$ corresponding to the six synthetic phantoms displayed in figure 5. Since ART and iNETT are both iterative approaches, the artifacts in the images generated by ART is consistent with the artifacts to be regularized in the iNETT algorithm.

We separate the 2600 prepared samples into three sets: the training set of 2000 samples, $\{(z_s, r_s) \mid s \in \{1, \ldots, 1000\} \cup \{1301, \ldots, 2300\}\}$, the validation set of 400 samples, $\{(z_s, r_s) \mid s \in \{1001, \ldots, 1200\} \cup \{2301, \ldots, 2500\}\}$, and the test set of 200 samples, $\{(z_s, r_s) \mid s \in \{1201, \ldots, 1300\} \cup \{2501, \ldots, 2600\}\}$. Each set consists of two types of input images: half of images with artifacts, and half of artifact-free images. The convex U-net $\Phi_c^\Theta$ is trained according to Algorithm 1, where the regularization parameter $\lambda$ is taken as $5 \times 10^{-4}$, the batch size is set as $n_0 = |I_c| = 10$, and the learning rate is $5 \times 10^{-4}$. Figure 7 shows the convergence plot in the training process, where the blue curve illustrates the convergence of $L_c(\Theta)$ on training set, and the red curve shows its performance on validation set. Since we employ a mini-batch optimization approach with random shuffling, and the non-negativity constraint is imposed after every iteration, the cost function $L_c(\Theta)$ has an oscillating behavior. Finally, we evaluate
Figure 7. Convergence plot of $L_i(\Theta)$ in the training process. The blue curve illustrates the convergence on training set, and the red curve shows the performance on validation set.

Figure 8. Mean squared error (MSE) on the test set of 200 samples. The first 100 samples correspond to input images with artifacts, and the second 100 samples correspond to artifact-free images with $r_s = 0$. (a) MSE for the convex U-net $\Phi^c_{\Theta}$; (b) MSE for the general U-net $\Phi^s_{\Theta}$.

the trained convex U-net on the test set of 200 samples. The mean squared error (MSE) is used to measure the performance of reconstructions, and it is defined in the following way,

$$MSE = \frac{1}{N} \| \Phi^c_{\Theta} (z_s) - r_s \|_2^2, \quad z_s, r_s \in \mathbb{R}^N.$$  

Figure 8(a) shows the values of MSE for the trained convex U-net on the 200 test samples. In next subsection, we will illustrate the performance of iNETT and compare it with the standard NETT algorithm [37]. Since the NETT algorithm does not require convexity of the neural network, we train a general U-net without the constraint of convexity, denoted as $\Phi^s_{\Theta}$. The training set, validation set, and test set are taken the same as those of convex U-net $\Phi^c_{\Theta}$. Figure 8(b)
Figure 9. Example 1: synthetic phantom. (a) True model of attenuation image; (b) measurement data with 5% Gaussian noises; (c) reconstruction result by \textit{iNETT}; (d) reconstruction result by ART.

shows the values of MSE for the general U-net on the 200 test samples. It concludes that the performance of the convex U-net $\Phi_\Theta$ is almost comparable with that of the general U-net $\Phi_\Theta$ on the test set.

6.2. Reconstruction results

The \textit{iNETT} algorithm is implemented according to section 5.2. With the convex U-net $\Phi_\Theta$, we build the uniformly convex regularizer $R(x)$ for \textit{iNETT} as shown in section 5.3, where we set $a = 10^{-3}$. The sequence of regularization parameters $\alpha_n$ is taken as $2^{-n}$, $n \in \mathbb{N}^+$; the initial guess is $x_0 = \frac{1}{2} N$ with $N = 256^2$, and $\xi_0 \in \partial R(x_0)$. The stopping rule is the discrepancy principle as shown in equation (5.1), where we take the constant $\tau = 1.01$.

6.2.1. Example 1: synthetic phantom with non-overlapping ellipses. Figure 9(a) shows the true model of attenuation image. It is a synthetic phantom of the same type as the phantom images in the training set and validation set, but it is not contained in those phantom images.

Figure 9(b) shows the simulated measurement data, where we add 5% Gaussian noises with zero mean and unit variance. The \textit{iNETT} algorithm is performed to reconstruct the attenuation image from the noisy measurement data. Figure 9(c) shows the reconstruction result. Figure 9(d) provides the reconstruction result by ART, where we have performed 5 rounds of
Table 1. (Example 1) PSNR and SSIM for the reconstruction results by iNETT, ART, SIT and NETT, respectively.

|                | iNETT | ART  | SIT  | α = 0.001 | α = 0.01 | α = 0.05 | α = 0.1 |
|----------------|-------|------|------|-----------|----------|----------|---------|
| PSNR           | 21.30 | 14.48| 17.97| 16.85     | 19.34    | 18.67    |
| SSIM           | 0.84  | 0.54 | 0.66 | 0.62      | 0.79     | 0.79     |

Notes: In bold are highlighted the best values for each metric.

Figure 10. Example 1: synthetic phantom. Reconstruction result by the standard iterated Tikhonov (SIT) method.

ART iterations to get the solution. To quantitatively evaluate the performance of reconstruction, we compute the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM), where for both a higher value means a better reconstruction (note that the range of SSIM is between 0 and 1); the results are shown in the second and third columns of Table 1. The iNETT algorithm provides much better reconstruction for the synthetic phantom. It is capable of removing artifacts due to under-sampling and data noises while preserving the resolution of ellipses in the reconstructed image.

As a comparison, we provide reconstruction results by a standard iterated Tikhonov (SIT) method and by the NETT algorithm [37], respectively. In the SIT method, the regularizer $R$ is taken as $R(x) = ||x||_2^2$ without the convex neural network in iNETT. The iteration is the same as that of iNETT, while it has a closed form in this standard situation:

$$x^\delta = x_{n-1}^\delta - (F^T F + \hat{\alpha}_n I)^{-1} F^T (F x_{n-1}^\delta - y^\delta),$$  

(SIT)

where $F^T$ denotes the transpose of $F$, $I$ is the identity matrix, and $\hat{\alpha}_n = 2 M \alpha_n$; recall that $\alpha_n = 2^{-n}$, and $M = 256 \times 60$. The initial guess and the stopping rule are the same as those of iNETT. Figure 10 shows the reconstruction result by SIT, and Table 1 lists the values of PSNR and SSIM for this reconstruction. It shows that the SIT algorithm has a better performance than ART, but the improvement is inadequate comparing to the performance of iNETT.

In the NETT algorithm, the attenuation image is recovered by solving the following minimization problem:

$$x^\delta = \arg\min_{x \in X} \frac{1}{2M} ||Fx - y^\delta||_2^2 + \alpha \|\Phi_\Theta(x)\|_2^2,$$

NETT
where $\Phi_{\Theta}$ denotes the general U-net trained in section 6.1, and $\alpha$ is a fixed parameter controlling the amount of regularization. As stated above, $\Phi_{\Theta}$ is trained without the constraint of convexity, since the NETT algorithm does not require convexity of neural network in general. We perform the NETT algorithm with a series of different values of $\alpha$, although we are not trying to tune the parameter $\alpha$ exhaustively. Figure 11 shows the reconstruction results with $\alpha = 0.001, 0.01, 0.05$ and $0.1$, respectively. Table 1 lists the values of PSNR and SSIM for those reconstruction results. It shows that the NETT algorithm achieves the best reconstruction among the four solutions as $\alpha = 0.05$. Comparing the results in figures 9(c) and 11, and considering the values of PSNR and SSIM listed in table 1, we conclude that the iNETT algorithm can achieve a comparable reconstruction as NETT without the procedure of tuning the regularization parameter manually.

### 6.2.2. Example 2: synthetic phantom with overlapping ellipses

Figure 12(a) shows the true model of attenuation image. Again, it is a synthetic phantom of the same type as the phantom images in the training and validation sets, but it is not contained in them.

Figure 12(b) shows the simulated measurement data, where we add 5% Gaussian noises with zero mean and unit variance. Figure 12(c) shows the reconstruction result by the iNETT algorithm. Figure 12(d) provides the reconstruction result by ART, where we have performed five rounds of ART iterations to get the solution. To quantitatively evaluate the performance of reconstruction, we compute the values of PSNR and SSIM as well; the results are shown in the
Figure 12. Example 2: synthetic phantom. (a) True model of attenuation image; (b) measurement data with 5% Gaussian noises; (c) reconstruction result by iNETT; (d) reconstruction result by ART.

Table 2. (Example 2) PSNR and SSIM for the reconstruction results by iNETT, ART, SIT and NETT, respectively.

|            | iNETT | ART  | SIT  | NETT α = 0.001 | NETT α = 0.01 | NETT α = 0.05 | NETT α = 0.1 |
|------------|-------|------|------|----------------|---------------|---------------|--------------|
| PSNR       | 25.35 | 17.26| 21.85| 19.51          | 22.46         | 22.36         | 21.66        |
| SSIM       | 0.91  | 0.63 | 0.77 | 0.70           | 0.81          | 0.84          | 0.83         |

Notes: In bold are highlighted the best values for each metric.

second and third columns of table 2. It shows that the iNETT algorithm provides much better reconstruction for the synthetic phantom. It is capable of removing artifacts in the reconstructed image and achieving higher values of PSNR and SSIM.

As a comparison, we provide reconstruction results by SIT and NETT, respectively. The solution by SIT is shown in figure 13, and the values of PSNR and SSIM are listed in the fourth column of table 2. The SIT algorithm has a better performance than ART, but the improvement is inadequate comparing to the performance of iNETT. The reconstruction results by NETT are shown in figure 14, where we perform the NETT algorithm with $\alpha = 0.001, 0.01, 0.05$ and 0.1, respectively. The PSNR and SSIM of these reconstruction results are listed in table 2. Comparing the results in figures 12(c) and 14, and considering the values of PSNR and SSIM listed in table 2, we conclude that the iNETT algorithm can achieve a comparable (if not better) reconstruction as NETT without tuning the regularization parameter exhaustively.
Figure 13. Example 2: synthetic phantom. Reconstruction result by the standard iterated Tikhonov (SIT) method.

Figure 14. Example 2: synthetic phantom. Reconstruction results by NETT with $\alpha = 0.001, 0.01, 0.05$ and 0.1, respectively. (a) $\alpha = 0.001$; (b) $\alpha = 0.01$; (c) $\alpha = 0.05$; (d) $\alpha = 0.1$.

6.2.3. Example 3: attenuation images of lung and myocardium. In this example, we consider reconstructions of attenuation images which are quite different from the synthetic phantoms in the training process.
Figure 15. Example 3: lung imaging. (a) Attenuation image of lung; (b) measurement data with 5% Gaussian noises; (c) reconstruction result by iNETT; (d) reconstruction result by ART.

The true models of attenuation images are shown in figures 15(a) and 16(a), respectively, where figure 15(a) shows the attenuation image of lung, and figure 16(a) shows the attenuation image of myocardium. In the iNETT algorithm, we employ the same convex U-net $\Phi^c_\Theta$ trained on synthetic phantoms as described in section 6.1. The reconstruction results of lung imaging are provided in figure 15, where figure 15(b) shows the measurement data with 5% Gaussian noises, figure 15(c) shows the recovered solution of iNETT, and figure 15(d) shows the recovered solution of ART. Correspondingly, the reconstruction results of myocardium imaging are provided in figure 16.

In addition, the values of PSNR and SSIM for these reconstruction results are listed in table 3. It shows that the iNETT algorithm provides much better solutions in both lung imaging and myocardium imaging, removing artifacts in the reconstructed images and achieving higher values of PSNR and SSIM. In practical applications, one can further improve the performance of iNETT by re-training the convex neural network $\Phi^c_\Theta$ according to the types of attenuation images to be reconstructed, e.g. using images of lungs to train $\Phi^c_\Theta$ in lung imaging and using images of myocardia to train $\Phi^c_\Theta$ in myocardium imaging. In this example, we use the convex U-net trained on synthetic phantoms for the reconstruction of lung and myocardium images, which is partly due to the lack of appropriate training data, and partly to illustrate the generalization ability of the iNETT algorithm.
Figure 16. Example 3: myocardium imaging. (a) Attenuation image of myocardium; (b) measurement data with 5\% Gaussian noises; (c) reconstruction result by iNETT; (d) reconstruction result by ART.

Table 3. (Example 3) List of PSNR and SSIM for the reconstruction results.

|                | lung imaging | myocardium imaging |
|----------------|--------------|--------------------|
|                | iNETT        | ART                | iNETT          | ART          |
| PSNR           | 20.02        | 16.46              | 19.95          | 16.08        |
| SSIM           | 0.70         | 0.51               | 0.69           | 0.46         |

Notes: In bold are highlighted the best values for each metric.

7. Conclusion

We have proposed a non-stationary iNETT method for the solution of ill-posed inverse problems. Motivated by the NETT method [37], we employ a data-driven regularization term including a neural network architecture, where the neural network is trained to penalize artifacts in the recovered solution. The iNETT method is then developed by combining the strategy of neural-network based regularization with the non-stationary iterated Tikhonov method. The main advantage of iNETT is that it avoids the difficult task of estimating the optimal regularization parameter, while keeps the virtue of the data-driven regularizer in NETT. We provide numerical examples to illustrate applications of iNETT in 2D computerized tomography. It shows that the iNETT method can achieve comparable (if not better) results as NETT without tuning the regularization parameter exhaustively.
To achieve the theoretical convergence of iNETT, we require that the neural-network based regularizer in iNETT is uniformly convex. As a result, we develop theories and algorithms for the construction of convex and uniformly convex neural networks. Given a general neural network, we prescribe sufficient conditions to achieve a related neural network which is component-wise convex or uniformly convex. Our formulation can embrace state-of-the-art convolutional neural networks. As a concrete example, we provide rigorous formulas for the U-net architecture, and explain the approaches to obtain convex and uniformly convex U-net architectures, which are successfully used in iNETT for the computations of 2D computerized tomography. The tool of convex and uniformly convex neural networks shall have many interesting applications in the future study.

For the iNETT method itself, there are several directions to further improve the algorithm. (1) Study more efficient neural network architectures with different data-fidelity norms, which can be more stable under random noises with non-Gaussian distributions, e.g. the $\ell^p$-$\ell^q$ norm regularization \[17\]. (2) Test alternative training strategies for the uniformly convex neural network regularizer, such as the adversarial strategy in \[40, 42\]. (3) Learn the sequence \$\{o_n\}\$ in iNETT, mimicking the procedure developed in \[10\]. (4) Devise nonlocal differential-like operators in the regularization term to enforce some a-priori knowledge on the recovered solution \[11, 14, 24\]. (5) Generalize iNETT to a fully learned network architecture, in the same spirit of \[52, 57\], e.g. learn the neural network regularizer $R(x)$ in the optimization process.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

Davide Bianchi is supported by NSFC (Grant No. 12250410253). Wenbin Li is supported by Natural Science Foundation of Shenzhen (Grant No. JCYJ20190806144005645) and NSFC (Grant No. 41804096). We thank the anonymous reviewers for their helpful and inspiring suggestions.

ORCID IDs

Davide Bianchi \(https://orcid.org/0000-0003-0635-0637\)
Wenbin Li \(https://orcid.org/0000-0002-1125-2364\)

References

[1] Aggarwal H K, Mani M P and Jacob M 2018 Modl: model-based deep learning architecture for inverse problems *IEEE Trans. Med. Imaging* 38 394–405
[2] Amos B, Xu L and Kolter J Z 2017 Input convex neural networks *Proc. 34th Int. Conf. on Machine Learning* (Proc. Machine Learning Research vol 70) ed D Precup and Y W Teh (PMLR) pp 146–55
[3] Andrews H C and Hunt B R 1977 *Digital Image Restoration* (Englewood Cliffs, NJ: Prentice-Hall)
[4] Antholzer S and Haltmeier M 2021 Discretization of learned nett regularization for solving inverse problems *J. Imaging* 7 1–13
[5] Antholzer S, Haltmeier M and Schwab J 2019 Deep learning for photoacoustic tomography from sparse data *Inverse Probl. Sci. Eng.* 27 987–1005
[6] Antholzer S, Schwab J, Bauer-Marschallinger J, Burgholzer P and Haltmeier M 2019 Nett regularization for compressed sensing photoacoustic tomography *Photons Plus Ultrasound: Imaging and Sensing* 2019 vol 10878 (San Francisco, CA: SPIE) pp 272–82

[7] Ao W, Li W and Qian J 2021 A data and knowledge driven approach for spect using convolutional neural networks and iterative algorithms *J. Inverse Ill-Posed Probl.* 29 543–55

[8] Arridge S, Maass P, Öktem O and Schönlieb C B 2019 Solving inverse problems using data-driven models *Acta Numer.* 28 1–174

[9] Bachmayr M and Burger M 2009 Iterative total variation schemes for nonlinear inverse problems *Inverse Problems* 25 105004

[10] Bertocchi C, Chouzenoux E, Corbineau M-C, Pesquet J-C and Prato M 2020 Deep unfolding of a proximal interior point method for image restoration *Inverse Problems* 36 034005

[11] Bianchi D, Buccini A, Donatelli M and Randazzo E 2022 Graph laplacian for image deblurring *ETNA* 55 169–86

[12] Bianchi D, Buccini A, Donatelli M and Serra-Capizzano S 2015 Iterated fractional Tikhonov regularization *Inverse Problems* 31 055005

[13] Bianchi D and Donatelli M 2017 On generalized iterated Tikhonov regularization with operator-dependent seminorms *ETNA* 47 73–99

[14] Bianchi D and Donatelli M 2021 Graph approximation and generalized Tikhonov regularization for signal deblurring 2021 21st Int. Conf. on Computational Science and Its Applications (ICCSA) (IEEE) pp 93–100

[15] Borwein J, Guirao A, Hájek P and Vanderwerff J 2009 Uniformly convex functions on banach spaces *Proc. Amer. Math. Soc.* 137 1081–91

[16] Buccini A, Donatelli M and Reichel L 2017 Iterated Tikhonov regularization with a general penalty term *Numer. Linear Algebra Appl.* 24 e2089

[17] Buccini A, Pasha M and Reichel L 2020 Modulus-based iterative methods for constrained $\ell_p-\ell_q$ minimization *Inverse Problems* 36 084001

[18] Cai Y, Donatelli M, Bianchi D and Huang T-Z 2016 Regularization preconditioners for frame-based image deblurring with reduced boundary artifacts *SIAM J. Sci. Comput.* 38 B164–89

[19] Chen H, Zhang Y, Zhang W, Liao P, Li K, Zhou J and Wang G 2017 Low-dose ct via convolutional neural network *Biomed. Opt. Express* 8 679–94

[20] Donatelli M 2012 On nondecreasing sequences of regularization parameters for nonstationary iterated Tikhonov *Numer. Linear Algebra Appl.* 19 93–122

[21] Dumoulin V and Visin F 2016 A guide to convolution arithmetic for deep learning (arXiv:1603.07285)

[22] Engl H W 1987 On the choice of the regularization parameter for iterated Tikhonov regularization of ill-posed problems *J. Approx. Theory* 49 55–63

[23] Engl H W, Hanke M and Neubauer A 1996 *Regularization of Inverse Problems* vol 375 (Dordrecht: Kluwer)

[24] Gilboa G and Osher S 2009 Nonlocal operators with applications to image processing *Multiscale Model. Simul.* 7 1005–28

[25] Gordon R, Bender R and Herman G T 1970 Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography *J. Theoret. Biol.* 29 471–81

[26] Hanke M 1996 Limitations of the L-curve method in ill-posed problems *BIT* 36 287–301

[27] Hanke M and Groetsch C W 1998 Nonstationary iterated Tikhonov regularization *J. Optim. Theory Appl.* 98 37–53

[28] Hansen P 1992 C Analysis of discrete ill-posed problems by means of the L-curve *SIAM Rev.* 34 561–80

[29] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. Ieee Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (Cvpr)* (Los Alamitos, CA: IEEE Computer Society) pp 770–8

[30] Heinrich M P, Stille M and Buzug T M 2018 Residual U-net convolutional neural network architecture for low-dose CT denoising *Curr. Dir. Biomed. Eng.* 4 297–300

[31] Higham C F and Higham D J 2019 Deep learning: an introduction for applied mathematicians *SIAM Rev.* 61 860–91

[32] Huang G, Liu Z, Van Der Maaten L and Weinberger K Q 2017 Densely connected convolutional networks *Proc. Ieee Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (Cvpr)* (Los Alamitos, CA: IEEE Computer Society) pp 4700–8
[33] Ioffe S and Szegedy C 2015 Batch normalization: accelerating deep network training by reducing internal covariate shift Int. Conf. on Machine Learning (PMLR) pp 448–56
[34] Jin Q and Stals L 2012 Nonstationary iterated Tikhonov regularization for ill-posed problems in banach spaces Inverse Problems 28 104011
[35] Jin Q and Zhong M 2014 Nonstationary iterated Tikhonov regularization in banach spaces with uniformly convex penalty terms Numer. Math. 127 485–513
[36] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[37] Li H, Schwab J, Antholzer S and Haltmeier M 2020 NETT: solving inverse problems with deep neural networks Inverse Problems 36 065005
[38] Li W, Wang K and Fan T 2022 A stochastic gradient descent approach with partitioned-truncated singular value decomposition for large-scale inverse problems of magnetic modulus data Inverse Problems 38 075002
[39] Lindenstrauss J and Tzafriri L 2013 Classical Banach Spaces II: Function Spaces vol 97 (Berlin: Springer Science and Business Media)
[40] Lunz S, Öktem O and Schönlieb C B 2018 Adversarial regularizers in inverse problems Advances in Neural Information Processing Systems vol 31
[41] Morotti E, Evangelista D and Loli Piccolomini E 2021 A green prospective for learned post-processing in sparse-view tomographic reconstruction J. Imaging 7 1–14
[42] Mukherjee S, Dittmer S, Shumaylov Z, Lunz S, Öktem O and Schönlieb C B 2020 Learned convex regularizers for inverse problems (arXiv:2008.02839)
[43] Obmann D, Nguyen L, Schwab J and Haltmeier M 2021 Augmented nett regularization of inverse problems J. Phys. Commun. 5 105002
[44] Obmann D, Schwab J and Haltmeier M 2020 Deep synthesis network for regularizing inverse problems Inverse Problems 37 015005
[45] Osher S, Burger M, Goldfarb D, Xu J and Yin W 2005 An iterative regularization method for total variation-based image restoration Multiscale Model. Simul. 4 460–89
[46] Reichel L and Rodriguez G 2013 Old and new parameter choice rules for discrete ill-posed problems Numer. Algorithms 63 65–87
[47] Rockafellar R T and Wets R J B 1998 Variational Analysis (Grundlehren der Mathematischen Wissenschaften vol 317) corrected 3rd printing 2009 edn (Berlin: Springer)
[48] Ronneberger O, Fischer P and Brox T 2015 U-net: Convolutional networks for biomedical image segmentation Proc. Med. Image Comput. Comput. Assist. Interv (Berlin: Springer) pp 234–41
[49] Safran I and Shamir O 2020 How good is SGD with random shuffling? Proc. 33rd Conf. on Learning Theory (Proc. Machine Learning Research vol 125), ed J Abernethy and S Agarwal (PMLR) pp 3250–84
[50] Scherzer O, Grasmair M, Grossauer H, Haltmeier M and Lenzen F 2009 Variational Methods in Imaging (Berlin: Springer)
[51] Sivaprasad S, Singh A, Manwani N and Gandhi V 2021 The curious case of convex neural networks Joint European Conf. on Machine Learning and Knowledge Discovery in Databases (Springer) pp 738–54
[52] Sun J, Li H and Xu Z et al 2016 Deep admn-net for compressive sensing MRI Advances in Neural Information Processing Systems vol 29
[53] Tan H Y, Mukherjee S, Tang J and Schönlieb C B 2022 Data-driven mirror descent with input-convex neural networks (arXiv:2206.06733)
[54] Wei Z and Chen X 2018 Deep-learning schemes for full-wave nonlinear inverse scattering problems IEEE Trans. Geosci. Remote Sens. 57 1849–60
[55] Zălinescu C 1983 On uniformly convex functions J. Math. Anal. Appl. 95 344–74
[56] Zălinescu C 2002 Convex Analysis in General Vector Spaces (Singapore: World scientific)
[57] Zhang J and Ghanem B 2018 Ista-net: interpretable optimization-inspired deep network for image compressive sensing Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 1828–37