Learning Sparsity-Promoting Regularizers using Bilevel Optimization
Avrajit Ghosh∗, Michael T. McCann†, Madeline Mitchell‡, and Saiprasad Ravishankar§

Abstract. We present a gradient-based heuristic method for supervised learning of sparsity-promoting regularizers for denoising signals and images. Sparsity-promoting regularization is a key ingredient in solving modern signal reconstruction problems; however, the operators underlying these regularizers are usually either designed by hand or learned from data in an unsupervised way. The recent success of supervised learning (e.g., with convolutional neural networks) in solving image reconstruction problems suggests that it could be a fruitful approach to designing regularizers. Towards this end, we propose to denoise signals using a variational formulation with a parametric, sparsity-promoting regularizer, where the parameters of the regularizer are learned to minimize the mean squared error of reconstructions on a training set of ground truth image and measurement pairs. Training involves solving a challenging bilevel optimization problem; we derive an expression for the gradient of the training loss using the closed-form solution of the denoising problem and provide an accompanying gradient descent algorithm to minimize it. Our experiments with structured 1D signals and natural images indicate that the proposed method can learn an operator that outperforms well-known regularizers (total variation, DCT-sparsity, and unsupervised dictionary learning) and collaborative filtering for denoising.

Key words. Sparse representations, denoising, analysis operator learning, transform learning, bilevel optimization, machine learning.

AMS subject classifications. 68W65, 68U65.

1. Introduction. Sparse representations have been widely used in image processing and imaging. In this paper, we develop an approach for supervised learning of nonsmooth sparsity promoting regularizers for image restoration. We first review several categories of image restoration or reconstruction methods from the literature, before summarizing our contributions.

1.1. Image Reconstruction. Image reconstruction deals with recovering an image from its noisy measurements, which has applications in medicine (X-ray CT [18,26], MRI [19,29,41,55], PET [4,45], SPECT [34]), remote sensing [10,50], astronomical imaging [3,32], etc. Image reconstruction falls under the broad category of inverse problems, which deal with recovering the underlying information from incomplete observations or measurements. A signal \( x \) can undergo some process to form measurements \( \mathbf{y} \). Let the process be modeled as an operator \( A(\cdot) \), i.e., \( \mathbf{y} = A(x) + \mathbf{e} \), with \( \mathbf{e} \) being measurement noise. Then, given \( \mathbf{y} \) and knowledge of what the process \( A(\cdot) \) is, we are interested in recovering \( x \). In many applications, this inverse operation is ill-posed, i.e., there can exist infinitely many solutions \( x \) that are consistent with the forward operator and the measurements. Hence, we need some prior knowledge about \( x \)
so that we choose meaningful solutions from the set of solutions that are consistent in some sense with the forward model. To phrase it in a formal way, we want to regularize the recovery problem in such a way that the solutions that fit our prior knowledge of $x$ would be preferred over the solutions which do not fit the prior knowledge. Herein, comes the importance of the term regularization.

To preserve the consistency of the measurements with the physical forward model, one often minimizes the data consistency term $\|A(x) - y\|_2^2$. Moreover, we design a functional $R(x)$, such that minimizing it would restrict the solution space based on our prior knowledge of the data. Thus, the effective objective is $J(x) = \|A(x) - y\|_2^2 + R(x)$, where $x \in \mathbb{R}^n$ is the underlying image to be recovered, $y \in \mathbb{R}^m$ are its noisy measurements, $A : \mathbb{R}^n \to \mathbb{R}^m$ is a measurement (forward) operator, and $R : \mathbb{R}^n \to \mathbb{R}$ is a regularization functional. So we solve the optimization problem:

$$\hat{x} = \arg \min_x \frac{1}{2} \|A(x) - y\|_2^2 + R(x)$$

1.2. Types of Regularizers. Characterizing $R(x)$ to obtain a good reconstructor is a critical step in image reconstruction.

An important regularization model is sparsity, popularized by advances in compressed sensing [17]. Sparsity of a signal or an image $x$ is the number of nonzeros in the signal (viewed relative to its dimension) obtained via the $\ell_0$ "norm" $||x||_0$. Minimizing $J(x) = \|Ax - y\|_2^2$ subject to an $\ell_0$ constraint $||x||_0 \leq s$ is an NP-hard problem. The popular convex relaxation of this nonconvex problem involves relaxing the $\ell_0$ "norm" to the $\ell_1$ norm, which has been proven to preserve sparsity.

Typically reconstructed images are expected to have certain well-known properties like smoothness, piecewise-continuity, or even local or nonlocal structural similarity. Rather than assuming the image is directly sparse, it is a common approach to find a sparse solution in a transform domain, i.e., we minimize $||Wx||_1$, where the $W$ matrix projects the image to a domain, where we expect it to be sparse. For example, $W$ could be the finite difference matrix (this case is also known as total variation (TV) minimization) for piecewise-constant signals, or if the image is relatively smooth, $W$ can be the discrete cosine transform (DCT) or the discrete wavelet transform (DWT) matrix depending on the nature of smoothness.

In all of the above methods, the transform matrix $W$ is known before-hand and hence the form of the regularizer is known. There has been much interest in the past decade to adapt the regularizer to data [41–43, 47]. The regularization parameters such as the sparsifying transform could be learned from a dataset of high-quality images or even directly from measurements [41] (during image reconstruction). These methods are a type of unsupervised learning [2, 41, 42, 44]. Supervised learning [25, 27, 38, 46, 53, 56, 58] uses paired training data (typically ground truth images and their corrupted measurements) to learn the regularizer.

1.3. Contributions. Learning sparsifying transforms in an unsupervised manner has been explored in several works. In recent times, supervised methods in image reconstruction have gained much attention. Methods like Plug and play Priors (PnP) [54] and Regularization by denoising (RED) [46] have been increasingly popular because of their flexibility and enhanced

---

1 The $\ell_0$ functional is not really a norm because it is not homogeneous.
Bilevel Learning of $\ell_1$ Regularizers

performance for image reconstruction. Although researchers have focused a lot of attention on analyzing the convergence of PnP and RED [9,11,33], plugging a black-box denoiser into an iterative algorithm may not have significant mathematical interpretation. Therefore, we propose a novel way of learning a sparsifying transform in a supervised fashion using paired training data to function within a variational formulation for denoising.

Consider the process where a signal/image is contaminated by noise to form noisy measurements. Suppose there are $T$ pairs of such clean signals and measurements denoted by $(x_t, y_t)_{t=1}^T$. The measurements ($y_t \in \mathbb{R}^n$) are related to the clean signal ($x_t \in \mathbb{R}^n$) as $y_t = x_t + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. We propose learning a sparsifying transform transform $W \in \mathbb{R}^{n \times n}$ for denoising using the $T$ sets of paired data $(x_t, y_t)_{t=1}^T$ via the following formulation:

\[
\begin{align*}
(1.2a) \quad \arg\min_W Q(W) &= \frac{1}{T} \sum_{t=1}^T \frac{1}{2} \|x_t^*(W, y_t) - x_t\|_2^2 \\
(1.2b) \quad \text{s.t.} \quad x_t^*(W, y_t) &= \arg\min_x \|x - y_t\|_2^2 + \|Wx\|_1.
\end{align*}
\]

The intuition for learning such a transform $W$ would be: If (1) $x_t^*(W, y_t)$ is the result of denoising of $y_t$ using a transform $W$, then (2) the result $x_t^*(W, y_t)$ should be close to the ground-truth $x_t$. These two statements are formulated mathematically as the lower-level problem and the upper-level problem respectively, in the above bilevel optimization. Statement (1) is cast as a lower-level denoising problem to denoise $y_t$ using transform $W$. Statement (2) makes sure that the result of the lower-level problem $x_t^*(W, y_t)$ is close to the ground-truth signal $x_t$ in an $\ell_2$ norm sense (this could be replaced in general with other quality metrics).

We solve this bilevel optimization problem using our proposed algorithm which we term as "Bilevel Learning of $\ell_1$ regularizers with Closed-Form Gradients" dubbed BLORC. Before conveying our approach of solving the optimization problem, we give a brief review of the existing methods of solving such bilevel optimization problems (Section 2). In doing so, we point out the drawbacks of the existing algorithms which attempt to learn regularizers in bilevel optimization of the form (1.2). A recent work of ours in [21] presented a very brief overview of our proposed approach along with some preliminary results. We extend the work here in terms of theory, analysis, derivation, and detailed experiments. We considered the denoising problem in this work to derive and demonstrate our approach and we believe the methods could be extended to other inverse problems as well. We summarize our contributions as follows:

1. We propose a simple heuristic approach for (1.2) by replacing the lower-level problem (1.2b) with a derived closed-form expression.
2. We provide a detailed analysis of the nature of a sign-pattern (arising from the closed-form expression), which allows us to obtain explicit gradient expressions in local neighborhoods of the transform, hence making gradient descent possible.
3. We address the drawbacks of the existing methods for solving (1.2) which are discussed in detail in Section 2.5 and we discuss how our method overcomes these drawbacks.
4. We perform experiments on image denoising on some images from Urban-100 Dataset and compare with methods like BM3D and unsupervised sparsifying operator learning methods like Analysis K-SVD [47]. To the best of our knowledge, our image denoising
experiments are the first that involves learning sparsifying transforms by solving bilevel optimization with a non-smooth lower level cost.

We note here that we are applying gradient descent to a function that is not differentiable everywhere (piecewise differentiable); while this is potentially theoretically concerning, it is common in practice, e.g. [20, 30, 31]. We discuss this in more details in Section 3.2. Our experiments indicate that our method works well in practice. In this paper, we present a heuristic method and the drawn conclusions are mainly based on empirical observations.

1.4. Organization. The remainder of the paper is organized as follows. Section 2 briefly summarizes the well-known approaches to tackle a bilevel optimization problem and how our proposed approach BLORC performs when compared to these approaches. Section 3 discusses in detail the methods we propose to solve the bilevel optimization problem, which form the bedrock of our algorithm presented in Section 4. Experiments performed on 1D signals and 2D images using the BLORC algorithm are presented in Section 5. Section 6 discusses how BLORC performs under various underlying noise levels and under what circumstances its performance breaks down. Section 7 presents the mathematical proofs and derivations of results presented in the methods section. Finally, in Section 8, we conclude.

2. Bilevel optimization in inverse problems. Image reconstruction has been studied extensively under the lens of bilevel optimization [16, 28]. In this section, we provide a brief overview of some well-known methods for solving such an optimization problem. Without loss of generality, we rewrite the bilevel optimization problem presented in (1.2) in a more general form as follows:

\[
\hat{\beta} = \arg \min_{\beta} \mathcal{L}(\beta, x^*(\beta))
\]

s.t. \(x^*(\beta) = \arg \min_x \mathcal{G}(x, \beta)\).

It comprises two optimization problems. The upper level problem minimizes the function \(\mathcal{L}(\beta, x^*(\beta))\). This loss function has an argument \(x^*(\beta)\) which comes from solving a lower level optimization problem by minimizing \(\mathcal{G}(x, \beta)\). Here, \(\beta\) denotes optimizable parameters of the lower-level problem. In the general form, we represent the optimization variable as the vector \(\beta\) instead of a matrix \(W\) unlike in (1.2). It is straightforward to extend all the derivations when the learnable parameter is a matrix instead of a vector.\(^2\) We refer the readers to the review article by Crockett et al. [14], which summarizes in more detail the existing literature on bilevel optimization for image reconstruction.

In bilevel optimization problems, the variable \(\beta\) connects both the upper level and the lower level problems. It is easy to see that if \(x^*(\beta)\) can be explicitly written as a function of \(\beta\), then the bilevel problem can be converted into a typical single level optimization problem, which can be easily tackled by optimization techniques using the chain rule of differentiation.

\(^2\)Note that for a matrix \(W\), (2.1) can be rewritten as \(\nabla \mathcal{L}(W) = \nabla_W \mathcal{L}(W, x^*) + (\nabla_W x^*(W))\nabla_x \mathcal{L}(W, x^*(W))\). In this case, \(\nabla \mathcal{L}(W)\), \(\nabla_W \mathcal{L}(W, x^*)\) are matrices and \((\nabla_W x^*(W))\) is an order 3 tensor and the vector \(\nabla_{x^*} \mathcal{L}(W, x^*(W))\) is multiplied to a dimension of the tensor. Hence, the product of the order 3 tensor and the vector \((\nabla_W x^*(W))\nabla_x \mathcal{L}(W, x^*(W))\) is a matrix. The product of the order 3 tensor \((T)\) and a vector \(x\) is a matrix and is defined as \(\sum T(i, j, k)x(i)\).
But this is rarely the case. For learning sparsifying transforms in a bilevel framework, the upper-level cost \( L \) is smooth and convex w.r.t. the reconstructions but the lower-level cost \( G \) is convex but non-smooth and non-differentiable. For these problems, \( x^*(\beta) \) can’t be written explicitly in terms of \( \beta \). Assuming the function \( G(x, \beta) \) is a strictly convex function of \( x \) (hence the lower level optimization problem has a unique minimizer), then using chain rule, we have

\[
(2.1) \quad \nabla L(\beta) = \nabla_\beta L(\beta, x^*) + \left( \nabla_\beta x^*(\beta) \right)^T \nabla_{x^*} L(\beta, x^*(\beta)).
\]

In our problem, where \( G \) is nonsmooth, we do not have an explicit expression for \( x^*(\beta) \) in terms of \( \beta \). Hence obtaining the term \( \nabla_\beta x^*(\beta) \) (also termed as implicit gradient) is the main challenge in solving such bilevel problems. The methods to derive \( \nabla_\beta x^*(\beta) \) can be broadly classified in four categories as briefly discussed in Sections 2.1, 2.2, 2.3, and 2.4. The derivation of the implicit gradient \( (\nabla_\beta x^*(\beta)) \) using these methods follows closely with [14].

2.1. Implicit Function Theorem (IFT). In IFT, we assume that the lower level solution \( (x^*(\beta)) \) can be expressed as an implicit function. Also, we assume that the lower level is unconstrained. Then there must exist a minimizer of \( G \) such that the gradient with respect to lower level optimization variable \( x \) at the minimizer \( x^* \) is zero, i.e., \( \nabla_x G(x^*(\beta), \beta) = 0 \). Using the chain rule to differentiate both sides, we can obtain an expression for \( \nabla_\beta x^*(\beta) \) that when substituted in (2.1) yields:

\[
(2.2) \quad \nabla L(\beta) = \nabla_\beta L(\beta, x^*(\beta)) - \nabla_x \beta G(x^*, \beta)^T [\nabla_x G(x^*, \beta)]^{-1} \nabla_{x^*} L(\beta, x^*(\beta)).
\]

2.2. Using KKT conditions. Replacing the lower level problem by \( \nabla_x G(x^*(\beta), \beta) = 0 \), a similar expression (Equation (2.2)) for calculating the gradient of the loss with respect to \( \beta \) can be found by the method of forming a Lagrangian

\[
L(x, \beta, \gamma) = L(\beta, x) + \gamma^T \nabla_x G(x, \beta),
\]

and finding an expression for \( \gamma \) using the KKT stationarity condition \( \nabla_x L(x, \beta, \gamma) = 0 \).

In problems that involve learning sparsifying transforms, the lower-level cost function \( G \) is non-smooth and non-differentiable w.r.t. the parameters \( \beta \). Hence in all the above formulations, \( \nabla_\beta G(x^*, \beta) \) is not-well defined over the domain of \( \beta \) rendering all the above expressions for \( \nabla_\beta x^*(\beta) \) not well-defined in that whole domain.

One way to avoid such a problem is to make a smooth approximation of the function that involves \( \beta \) (in this paper \( W \)). One of the first works in this direction is of [40]. Here the bilevel problem was optimized for the case when the lower level problem had an analysis sparsity prior \( ||Wx||_1 \). A smooth approximation of the \( \ell_1 \) regularizer was made to make the lower level cost function smooth w.r.t. the dictionary coefficients. Then the parameters of the dictionary were learned using the minimizer approach using KKT conditions as discussed above.

Another work [57] learns both the \( \ell_1 \) norm-based analysis and synthesis operators using bilevel optimization with implicit differentiation. However, the non-smooth \( \ell_1 \) penalty in the lower level problem was replaced by differentiable penalty functions to employ implicit differentiation. In [35], the authors address a synthesis version of the bilevel problem, wherein
the reconstruction problem involves finding sparse codes, \( z \) such that \( \| x - Wz \| \) is small. This change from the analysis to synthesis formulation means that the optimization techniques used in [35] do not apply here. In [51], the authors derived gradients for a generalization of (7.1) by relaxing \( \| Wx \|_1 \) to \( \min_z \alpha \| Wx - z \|_2^2 + \| z \|_1 \). This approach gives the gradient in the limit of \( \alpha \to \infty \), however the expression requires computing the eigendecomposition of a large matrix. Therefore the authors use the relaxed version \( \alpha < \infty \) in practice. A brief preprint work [12] derived a gradient for the upper level cost \( Q(W) \) in equation (1.2) by expressing \( W \) in a fixed basis and using a differentiable relaxation. Finally, [13] provides a nice overview of the topic of analysis operator learning in its various forms, and also tackles the bilevel optimization problem in (1.2) using a differentiable sparsity penalty.

2.3. Unrolling and related approaches. The idea of unrolling approaches is to replace the lower level optimization problem by a sequence of iterations. Iterative algorithms have been used extensively to solve convex optimization problems. A very common optimization problem prevalent in image reconstruction is the unconstrained \( \ell_1 \) minimization problem which consists of an \( \ell_1 \) penalty and a data-fidelity term. ISTA (Iterative Shrinkage-Thresholding Algorithm) algorithm and ADMM (Alternating direction method of multipliers) are well known iterative algorithms to solve this type of problem. Say if the algorithm used to solve the lower level problem is unrolled for \( T \) iterations, then for each iterative update, let \( \Psi \) denote the update as a function, i.e.,

\[
x^{(t)} = \Psi(x^{(t-1)}, \beta) \quad \text{for} \ t = 1, 2, ..., T.
\]

(2.3)

Assuming the algorithm runs for \( T \) iterations, we will use \( x^{(T)} \) as the approximate reconstruction in the upper level loss. Let \( H_t = \nabla_x \Psi(x^{(t-1)}, \beta) \) and \( J_t = \nabla_{\beta} \Psi(x^{(t-1)}, \beta) \), then using chain rule yields

\[
\nabla_{\beta} x^*(\beta) \approx J_T + \sum_{t=1}^{T-1} (H_T H_{T-1} ... H_{t+1}) J_t.
\]

(2.4)

In the unrolling based approaches, a chain of sub-gradients are obtained for the unrolled iterations. For example, [39] differentiates the iterations of a non-linear primal primal-dual algorithm. Depending on the sequence order of calculating the sum in the implicit gradient (2.4), unrolling methods can be classified into forward and reverse mode. For the more common reverse mode [14], the gradient computation starts from the \( T \)th step to calculate the product of \( H_t \)'s. In each iteration, \( H_{t+1} \) and \( J_t \) are calculated. In doing so, for each iteration, the corresponding \( x^{(t)} \in \mathbb{R}^{n \times 1} \) is stored in memory and the matrices \( H_{t+1} \) and \( J_t \) are calculated and stored in memory.

Hence, the memory cost for the unrolling methods using the reverse mode becomes \( \mathcal{O}(Tn^2 + n^3) \). Also, unrolling for a large number of iterations may suffer from the problem of vanishing gradients.

As the reconstruction may not be accurate if the algorithm solving the lower-level problem is unrolled only for finite iterations, researchers have recently proposed Deep Equilibrium models (DEM) [22]. These models correspond to potentially infinite iterations of unrolling.
in the lower-level and hence may achieve better accuracy. DEM assumes that equation (2.3) has a fixed point $x^\infty$ such that a deep network or mapping $f$ with parameters $\theta$ can represent the fixed point as $x^\infty = f_\theta(x^\infty)$. The network parameters are learned to minimize the loss $L = \|x_t - x^\infty\|^2_2$ by an implicit differentiation of the fixed point equation $x^\infty = f_\theta(x^\infty)$. This helps convert the memory intensive task of backpropagating through many iterations of $f_\theta(.)$ to calculating an inverse Jacobian product, which can be further calculated by finding another approximate fixed point (solved by Neumann series). Hence, the task of finding the gradient of the loss $L$, i.e., $\frac{\partial L}{\partial \theta}$ just boils down to three major steps each iteration: a) calculating the residual, b) solving a fixed point equation by Neumann series (assuming it converges), and c) multiplying gradient of network output $\frac{\partial f_\theta(x^\infty)}{\partial \theta}$ with the fixed point solution obtained in (b). Similar to DEM, our approach BLORC can be thought of as unrolling the lower-level optimization to infinite iterations. This is because in BLORC, the lower-level is replaced by an implicit closed-form expression, just like the fixed point expression $x^\infty = f_\theta(x^\infty)$ in DEM is implicit. The difference is that we solve the lower-level problem in its exact form for learning the transform $W$ rather than relying on the mapping $f_\theta$ to represent generic fixed points.

2.4. Differentiating through a transformed cone problem (Diffcp) [1]. Every convex optimization problem including LASSO type problems (1.2b) can be converted to a cone program which deals with minimizing a linear function over the intersection of a subspace and convex cone. A solution map is defined to be a function mapping numerical data defining the problem to the primal or dual solution of the problem. For example, the lower-level problem can be written as an implicit solution map $G$ of the form $x^*(W) = G(y, \beta, W)$. The objective of differentiating through a cone problem is to calculate how the perturbations in the variable $(W)$ affect the solution $x^*(W)$, i.e., calculating $x^*(W + \delta W) = G(y, \beta, W + \delta W)$. This is done by decomposing the function $G$ into a composition of differentiable atomic functions and then representing the derivative of each atomic function as an abstract linear map [1] by a method called Automatic Differentiation (AD). For a LASSO type problem, there exist points in the numerical data domain $(W)$ on which the problem is non-differentiable. On these points, AD libraries compute heuristic quantities instead of derivatives [1]. The drawbacks of autodifferentiation methods over BLORC are:

1. In AD approaches, the task is divided into a sequence of differentiable operations as computational graphs on which backpropagation is performed through chain rule [36]. This division into a sequence of operations and calculating gradients for each node of the graph can utilize significant memory.\(^3\). This can be avoided if there is an analytical explicit expression for gradients.

2. Hand-coded analytical method of calculating gradient is known to be the most accurate method [7] as it calculates exact derivatives. Hand-coded gradient methods are also known to have less "Gradient to Function Compute Time Ratio" than AD methods (see Figure 1 in [7]). To further validate this, we perform an experiment in Section 5.3 to compare BLORC with the Diffcp implementation in terms of accuracy and time complexity.

\(^3\)For AD approaches, the memory requirement is dynamic, depending on the implementation of the expression graphs. Memory optimizing strategies [36] like retaping, checkpoints, and region-based memory aim to increase memory efficiency.
2.5. Drawbacks addressed by BLORC. The drawbacks of the methods discussed in Sections 2.1, 2.2, 2.3, and 2.4 are briefly summarised as follows:

1. Minimizer methods (KKT/IFT) to learn sparsifying transforms $W$ require calculating and storing the implicit gradient $\nabla_{W}x^*(W)$ to evaluate the gradient of the loss $\nabla L(W)$ as in the expression in the footnote 2. Storing the implicit gradient is of order $O(n^3)$.

2. Replacing the $\ell_1$ penalty by a smooth and differentiable approximator may result in corner-rounding. See Section 5.4 for comparison of $\ell_1$ penalty with smooth Huber loss.

3. Unrolling the iterations of an algorithm (e.g., ISTA) for solving the lower-level variational problem and then back-propagating gradients (w.r.t. $W$) through the iteration steps can suffer from storing quantities for every iteration.

4. Unrolling method approximates the solution of the lower level problem $x^*$ using a finite number of iterations $T$, hence calculating only an approximation of the implicit gradient $\nabla_{W}x^*(W)$.

The method we propose alleviates such drawbacks and we summarize this in Table 1.

| Methods            | Minimizer using KKT/IFT [12, 35, 40, 51, 57] | Unrolled methods [39] | BLORC (Ours) |
|--------------------|---------------------------------------------|------------------------|--------------|
| Memory             | $O(n^3)$                                    | $O(T + n^3)$           | $O(n^2)$     |
| Gradient Accuracy  | ×                                           | ×                      | ✓            |

Table 1: Comparison of some existing methods with our proposed approach BLORC to solve the bilevel optimization problem in (1.2). For minimizer methods, storing the implicit gradient has cost $O(n^3)$. For BLORC, only the gradient matrices require storing, so the memory cost is $O(n^2)$. We compare BLORC and Diffcp in Section 5.3.

3. Methods. To tackle the bilevel optimization problem, we proceed with the following steps:

1. An implicit closed-form expression of the lower-level problem is derived using KKT stationarity conditions. We will show both theoretically (in some special cases) and experimentally that in local neighborhoods of the transform, we get an explicit closed-form expression.

2. We take the differential of the locally explicit closed-form expression, and using matrix algebra [37], explicit gradient expressions of the upper-level loss w.r.t. the rows of the transform are obtained. This helps us to avoid explicitly calculating and storing the implicit gradient $\nabla_{W}x^*(W)$. Hence for BLORC, the memory cost is $O(n^2)$ (1) which involves just storing the gradient of upper-level loss ($\nabla_{W}Q$) each iteration.

3. Using the analytical gradient expression, mini-batch gradient descent is performed. It is important to note two things in the bilevel problem (1.2):
   a. No constraints are imposed on the matrix $W$ while learning. This lets us learn more
b We let the algorithm learn the scaling of the regularization penalty and hence we do not require a separate scalar regularization strength to be learned. \( (\beta \|Wx\|_1 = \|\beta Wx\|_1) \). Thus, the lower level problem does not contain a scalar regularization parameter.

3.1. Closed-form solution obtained by duality. Consider the lower-level functional

\[
J(x, W, y) = \frac{1}{2}\|x - y\|_2^2 + \|Wx\|_1,
\]

with \( x, y \in \mathbb{R}^n \) and \( W \in \mathbb{R}^{k \times n} \). It is strictly convex in \( x \) (because the \( \ell_2 \) norm term is strictly convex and the \( \ell_1 \) norm term is convex) and therefore has a unique global minimizer. Then we are justified in writing \( x^*(W) = \arg \min_x J(x, W, y) \) without the possibility of the minimizer not existing or being a nonsingleton. Note that although \( x^* \) depends on \( y \) and \( W \), the \( y \)-dependence is not relevant for this derivation and we will notate it explicitly. Our key observation in deriving the closed-form expression of \( (3.1) \) is that we need to know the sign pattern of \( Wx^*(W) \). So, the closed-form expression is an implicit equation where the reconstruction \( x^*(W) \) is itself dependent on \( \text{sign}(Wx^*) \) (where \( \text{sign}(z) \) is defined to be -1 when \( |z| < 0 \); 0 when \( |z| = 0 \); and 1 when \( |z| > 0 \)). Let \( k_0 \) denote the set \( \{i \in (1,2,3,..,k) : (Wx^*)_i = 0 \} \) and \( k_{\neq 0} \) denotes \( \{i \in (1,2,3,..,k) : (Wx^*)_i \neq 0 \} \). Also, let us define \( W_0 \) as the matrix containing the rows of \( W \) that are indexed by the set \( k_0 \) and \( W_\pm \) contains rows of \( W \) that are indexed by the set \( k_{\neq 0} \). Then we have the following theorem. A similar closed-form expression has been obtained in equation (33) in [52] using a different method.

**Theorem 3.1 (Closed-form expression for \( \arg \min_y J(x, W) \)).** Let the nonzero pattern \( s \) denote \( \text{sign}(Wx^*)W_{x^*\neq 0} \) and let \( W_0, W_\pm \) contain the rows of \( W \), whose indices are given by the sets \( k=0 \) and \( k_{\neq 0} \), respectively. Then the closed-form expression of the optimization problem \( x^* = \arg \min_x \frac{1}{2}\|x - y\|_2^2 + \|Wx\|_1 \) is obtained from Lagrangian dual analysis as

\[
x^*(W) = P_{N(W_0)}(y - W^T_\pm s),
\]

where \( P_{N(W_0)} \) is the projector matrix onto the nullspace of \( W_0 \) and is given by \( P_{N(W_0)} = (I - W_0^TW_0) \).

To obtain the closed-form expression for a particular point on the domain of \( W \), we solve the lower-level problem \( (1.2b) \) to obtain the reconstruction \( x^*(W) \). This is because \( W_\pm, W_0, s \) in \( (3.2) \) depend on \( x^*(W) \). One may ask the question of why should we use an implicit closed-form equation if we can find the reconstruction by methods like ADMM or PGD? But we re-emphasize that our objective in this step is not to find \( x^* \) but to obtain a closed-form expression that allows us to take gradients with respect to \( W \).

3.2. Local analysis of the closed-form expression. Generalized LASSO-type problems do not have an explicit closed-form solution [52]. In fact, the closed-form expression we derived in \( (3.2) \) is an implicit equation. Note that \( x^*(W) \) is a function of the sign pattern \( c(W) = \text{sign}(Wx^*(W)) \) because the row-splits \( W_0 \) and \( W_\pm \) are based on \( c(W) \). But again the sign pattern \( c(W) \) itself depends on \( x^*(W) \). The equation is implicit because \( x^*(W) \) appears
Thus considering each of the indices, the sign-pattern remains constant in a neighborhood of $x^*$ when $w=0$, $y$, $y$, $0$, and $w-|w|$ when $y-|w|\geq 0$ and 0 otherwise. As a result, $c((0, y)) = 1$, $c((-y, 0)) = -1$, and the equation (right-hand side) is explicit. For clarification, we do not assume this pattern is restricted to lie in a certain region, the dependency of the expression in (3.2) w.r.t. $W$, it is important to establish the fact there exists an $\epsilon$-neighborhood $\triangle W \in \mathbb{R}^{k \times n}$ such that $c(W + \triangle W) = c(W)$. Only then, we can treat the equation (3.2) as an explicit equation w.r.t. $W$ in an $\epsilon$-neighborhood of $W$. For clarification, we do not assume this pattern is piecewise constant except where $c((\infty, -y)\cup 0 \cup [y, \infty)) = 0$ is piecewise constant. A similar result holds when $w \leq 0$. Thus $Q(w) = (x^*(w) - x_t)^2$ is smooth except at $w = 0, -y, y$. Note that $Q(w)$ is differentiable in the intervals near the minimizers of $Q$ and the non-differentiable points form a set of zero measure.

### 3.2.1. The scalar denoising problem.

Consider the scalar denoising problem $x^*(w) = \arg\min_x \frac{1}{2}(x - y)^2 + wx$ and $c(w) = \text{sign}(wx^*(w))$. The measurement is assumed to be generated as $y = x_t + \epsilon$, where $x_t$ is the ground-truth. Assuming that $y \geq 0$, one can show that $x^*(w) = y - |w|$ when $y - |w| \geq 0$ and 0 otherwise. As a result, $c((0, y)) = 1$, $c((-y, 0)) = -1$, and $c((\infty, -y)\cup 0 \cup [y, \infty)) = 0$ is piecewise constant. A similar result holds when $w \leq 0$. Thus $Q(w) = (x^*(w) - x_t)^2$ is smooth except at $w = 0, -y, y$. Note that $Q(w)$ is differentiable in the intervals near the minimizers of $Q$ and the non-differentiable points form a set of zero measure.

### 3.2.2. Denoising problem under orthogonal constraint.

Consider the denoising problem $x^*_W(W) = \arg\min_{x} \frac{1}{2}\|x - y\|_2^2 + \lambda\|Wx\|_1$ but under an orthogonal constraint on (square) $W$ (lying on the Stiefel manifold), i.e., $WW^T = I$. It is easy to show that the solution to this optimization problem is given by $x^*_W(W) = W^TS_\lambda(Wy_i)$ where $S_\lambda(.)$ denotes the soft-thresholding function with parameter $\lambda$. Then the sign-vector $c_W(W) = \text{sign}(WW^T S_\lambda(Wy_i)) = \text{sign}(S_\lambda(Wy_i))$. Considering each of the indices, $[c_W(W)]_i$ is piecewise constant except where $|(Wy_i)_i| = \lambda$. Hence, $c_W(W)$ is piecewise constant in each of the coordinates except at points where $|(Wy_i)_i| = \lambda$. We show in Section 7.3 that these points form a set of Lebesgue measure zero over the

![Figure 1: The sign pattern $c(w) = \text{sign}(wx^*(w))$ is piecewise constant for the scalar denoising problem. Also, the upper-level cost $Q(w)$ is piecewise smooth with respect to $w$. Note that its non-differentiable points are also not local/global minimizers.](image)
The differential of the scalar Jacobian \( \nabla \text{derivative of scalar w.r.t. matrix} \) is the differential of a scalar with respect to a matrix. So, a general rule of thumb to calculate such derivatives is:

\[
W \partial W = \partial W \text{transform gradients of the upper level scalar loss of scalars with respect to matrices using differentials.}
\]

We briefly summarize how we obtain the gradients to exist for a specific training signal’s loss, the sign pattern vector \( \text{sign}(Wx^*) \) has to remain constant in an open set containing \( W \) for small \( \eta > 0 \). Thus, the upper level cost function \( Q(W) \) is valid in each region where \( \partial Q(W) = \text{sign}(Wx^*) \) is defined as

\[
\text{sign}(Wx^*) \quad \text{where} \quad x^* = \text{argmax}(Q(W)).
\]

In the experiments above, we chose the measurement dimension was \( 64 \times 64 \). Setting \( W = I \), the lower-level denoising problem is solved using a noisy signal \( y \) to obtain \( x^*(W) \) and hence using that to obtain \( c(W) = \text{sign}(Wx^*(W)) \).

2. Perturbing \( W \) in an \( \ell_\infty \) radius ball, we obtain the sign-pattern in a neighborhood of \( W \) as \( c(W + \Delta W) = \text{sign}((W + \Delta W)x^*(W + \Delta W)) \) such that \( \max(|\Delta W|_{ij}) \leq \eta \) for small \( \eta > 0 \). Our experiments showed that there always exists a \( \Delta W \) with any \( \eta = 10^{-8} \) (machine precision is \( 10^{-16} \)) for which \( c(W) = c(W + \Delta W) \). For a particular point on the domain of \( W \), we performed thousands of trials of generating \( \Delta W \) with \( \eta = 10^{-8} \), and every time \( c(W) = c(W + \Delta W) \) was satisfied. We also performed the experiment with \( W \) being a 1D-finite difference matrix, a zero matrix, and also a random matrix with its entries drawn from \( \mathcal{N}(0, 1) \) and also with different measurements (\( y \)) and observed similar results.

In the experiments above, we chose the measurement \( y \) to have length 64 and the matrix dimensions were \( 64 \times 64 \).

3.2.3. Experimental validation for constant sign pattern. In the generalized setting for \( W \), the denoising problem does not have a closed-form expression. So, to experimentally check our claim, we perform an experiment consisting of the following steps:

1. Setting \( W = I \), the lower-level denoising problem is solved using a noisy signal \( y \) to obtain \( x^*(W) \) and hence using that to obtain \( c(W) = \text{sign}(Wx^*(W)) \).
2. We then checked if \( c(W) = c(W + \Delta W) \). Our experiments showed that there always exists a \( \Delta W \) with any \( \eta = 10^{-8} \) (machine precision is \( 10^{-16} \)) for which \( c(W) = c(W + \Delta W) \). For a particular point on the domain of \( W \), we performed thousands of trials of generating \( \Delta W \) with \( \eta = 10^{-8} \), and every time \( c(W) = c(W + \Delta W) \) was satisfied. We also performed the experiment with \( W \) being a 1D-finite difference matrix, a zero matrix, and also a random matrix with its entries drawn from \( \mathcal{N}(0, 1) \) and also with different measurements (\( y \)) and observed similar results.

3.3. Gradient calculations. Next, we compute the gradient of the scalar \( Q(W) \) with respect to \( W \) using locally explicit closed-form expressions for the lower level problems. Second, for the gradients to exist for a specific training signal’s loss, the sign pattern vector \( \text{sign}(Wx^*) \) has to remain constant in an open set containing \( W \). Only then the closed-form expression for \( x^*(W) \) is valid in each region where \( \text{sign}(Wx^*) \) is constant.

We refer the readers to [37], which summarizes how to generate derivatives (Jacobian) of scalars with respect to matrices using differentials. We briefly summarize how we obtain gradients of the upper level scalar loss \( Q(W) \) with respect to sub-rows \( (W_0, W_\pm) \) of the transform \( W \) using differentials.

We denote the Jacobian of the scalar \( Q \) w.r.t. \( W \) as \( \nabla_W Q \), which has the same dimensions as \( W \) as it is just \( \left[ \frac{\partial Q}{\partial w_{ij}} \right] \) where \( w_{ij} \) is the element on the \( i \)th row and \( j \)th column of matrix \( W \). The differential of the scalar \( \partial Q(W) \) is defined as \( \partial Q(W) = Q(W + \Delta W) - Q(W) \), where \( \Delta W \) is the differential of the matrix \( W \), i.e., \( \Delta W = [\partial w_{ij}]_{i,j} \), and has the same dimension as \( W \).

From [37], we see that if we can write \( Q(W + \Delta W) = Q(W) + \text{Tr}(E^T \partial W) \), then the Jacobian \( \nabla_W Q = E \). Note that the canonical form \( \text{Tr}(E^T \partial W) \) is important to take the derivative of a scalar with respect to a matrix. So, a general rule of thumb to calculate such derivative of scalar w.r.t. matrix is:

1. Compute the differential relating the scalar \( (\partial Q) \) and the matrix \( \partial W \).
2. Rewrite the result into the canonical form \( \partial Q(W) = \text{Tr}(E^T \partial W) \) to derive the Jacobian.
\[ \nabla_{W} Q = E. \]

For our problem, we don’t have a direct relationship between \( Q \) and \( W \) to get a differential relation like \( \partial Q(W) = \text{Tr}(E^{T} \partial W) \). But rather the relation between \( \partial Q \) and \( \partial W \) is connected by an intermediate differential variable \( \partial x^{*}(W) \). \( \partial x^{*}(W) \) is the differential of the reconstruction in the lower-level problem which is a function of the transform \( W \). Since we have already defined a locally explicit expression for \( x^{*}(W) \) in (3.2), we can relate \( \partial x^{*}(W) \) and \( \partial W \) using (3.2) as discussed next. The proof for the following theorem is discussed in Section 7.

**Theorem 3.2 (Differential of the closed form).** Let \( c(W) \) denote the sign pattern of the reconstructed signal in the transform domain \( W \). Then if \( c(W) = \text{sign}(W x^{*}) \) is a constant vector in an open neighbourhood containing \( W \), then the gradient of the closed form in (3.2) w.r.t. \( W \) exists and the differential form is given by

\[
\begin{align*}
\partial x^{*} &= -P_{N(W_{0})} \partial W_{\pm}^{T} s \\
\partial x^{*} &= -(W_{0}^{+} \partial W_{0} P_{N(W_{0})})^{T} (y - \beta W_{\pm}^{T} s),
\end{align*}
\]

where \( \partial x^{*} \), \( \partial W_{0} \), and \( \partial W_{\pm} \) are the differentials of \( x^{*} \), \( W_{0} \), and \( W_{\pm} \), respectively.

Note that the closed-form expression in (3.2) for \( x^{*}(W) \) is in terms of the row-submatrices. To obtain the sub-matrices \( W_{\pm} \) and \( W_{0} \) from \( W \), we need to have the corresponding sign vector. This is obtained by solving the lower level problem (1.2b) using the current transform \( W \). Note that the differential expression (3.3) is valid only in a neighborhood of the transform where the sign pattern \( c(W) = \text{sign}(W x^{*}) \) is constant w.r.t. \( W \).

To relate \( \partial Q \) with \( \partial W_{\pm} \) and \( \partial W_{0} \), recall that \( Q(W) = \frac{1}{2} \| x^{*}(W) - x_{t} \|^{2} \) (restricting for now to one term in the upper level loss). Hence the gradient of \( Q \) w.r.t. \( x^{*} \) is \( \nabla_{x^{*}} Q = (x^{*}(W) - x_{t}) \). The canonical form for derivative of a scalar with respect to a vector is given as \( \partial Q = \nabla_{x^{*}} Q^{T} \partial x^{*} \) [37]. Using the expression for \( \nabla_{x^{*}} Q \), we have \( \partial Q = (x^{*}(W) - x_{t})^{T} \partial x^{*} \). Substituting for \( \partial x^{*} \) from (3.3) and re-arranging into the canonical form, we can get two expressions of the form \( \partial Q = \text{Tr}(E^{T} \partial W_{\pm}) \) and \( \partial Q = \text{Tr}(E^{T} \partial W_{0}) \), from which the gradients w.r.t. \( W_{\pm} \) and \( W_{0} \) are straightforward. These are shown next and are derived in Section 7.2.

**Corollary 3.3.** Let \( Q(W) = \frac{1}{2} \| x^{*}(W, y_{t}) - x_{t} \|^{2} \) be the upper level cost function that is smooth with respect to the intermediate reconstruction \( x^{*}(W) \), then we derive an expression for the gradient of the cost \( Q(W) \) with respect to \( W_{0} \) and \( W_{\pm} \) as

\[
\begin{align*}
\nabla_{W_{\pm}} Q &= -s \nabla_{x^{*}} Q^{T} P_{N(W_{0})} \\
\nabla_{W_{0}} Q &= -(P_{N(W_{0})}(q \nabla_{x^{*}} Q^{T} + \nabla_{x^{*}} Q q^{T}) W_{0}^{T})^{T},
\end{align*}
\]

with \( q = y_{t} - W_{\pm}^{T} s \). Here, \( \nabla_{x^{*}} Q = (x^{*} - x_{t}) \).

\(^{4}\)When we use the term differential \( \partial Q \), \( \partial x^{*}(W) \) or \( \partial W \) (same dimensions as \( Q \), \( x^{*}(W) \), and \( W \), respectively), we just mean the incremental changes in the scalar, vector, or matrix. A relationship such as \( \partial Q = \text{Tr}(E^{T} \partial W) \) indicates how the scalar \( Q \) changes for incremental changes in \( W \). And when we use \( \nabla_{W} Q \), we mean the derivative of the scalar \( Q \) w.r.t. \( W \). And the derivative can be obtained from the differential by writing it in the canonical form. That is if we have a differential relation \( \partial Q(W) = \text{Tr}(E^{T} \partial W) \), then the derivative \( \nabla_{W} Q = E. \)
It is important that the sign pattern \( c(W) \) (and \( x^*(W) \) itself) is accurate for the analytical gradient expression to be correct. Hence, during gradient descent w.r.t \( W \), we first find \( x^*(W) \) by running a large number of iterations of ADMM using the current transform \( W \). This step is the bottleneck of our proposed algorithm.

4. **BLORC Algorithm.** We perform minibatch gradient descent to learn the transform matrix \( W \) in a supervised manner from training pairs \((x_t, y_t)\). In all our experiments, we start from \( W_0 = I_{n \times n} \), the identity matrix. At the start of each epoch during training, the training pairs are randomly shuffled to remove dataset bias. Each sample in a batch is processed as follows.

**Algorithm 4.1 BLORC**

**Input:** \( M \) pairs of training signals \((x_t, y_t)\) \( M \), each of dimension \( n \times 1 \), initial transform matrix \( W_0 = I_{n \times n} \), batch size \( B \), number of epochs \( E \), learning rate \( \alpha \), threshold \( \gamma \).

**Output:** Trained \( \hat{W} \)

**Pre-processing:** For inputs that are clean and noisy images of size \( N \times N \), extract patches of size \( \sqrt{n} \times \sqrt{n} \) with stride \( r \) and vectorize them.

**Initialization:** \( W = W_0 \).

**for each epoch do**

a) **for each training index in batch do**

1) Obtain sign vector:
\[
x^* = \arg \min_x \|y_t - x\|^2 + \lambda\|Wx\|_1 / \text{Solve iterations of ADMM to obtain } x^*
\]
\[
s = \text{sign}(Wx^*)[|Wx^*|]_{\geq \gamma} / \text{Obtain the sign vector}
\]

2) **Pre-computations for gradient:**
\[
\nabla_{x^*} Q = 2(x^* - x_t), q = y_t - W_Ts
\]
Split rows of \( W \) into \( W_\pm \) and \( W_0 \) based on \( \text{sign}(Wx^*) \)

3) **Gradient Calculation:**
\[
\nabla_{W_\pm} Q = -s(\nabla_{x^*} Q)^T P_{N(W_\pm)}mM / \text{Obtain gradient for } W_\pm
\]
\[
\nabla_{W_0} Q = -(P_{N(W_0)}(q(\nabla_{x^*} Q)^T + \nabla_{x^*} Qq)^T W_0^+)\] / \text{Obtain gradient for } W_0

4) **Stack the gradient matrices based on row partition**
\[
\nabla W = [\nabla_{W_0}Q; \nabla_{W_\pm}Q]
\]

5) **Accumulate the gradients \( \nabla W \) for the current batch.**

end

b) **Update \( W \) after each batch is processed.**
\[
W = W - \alpha \nabla W Q
\]

end

\( \hat{W} = W \)

**Post-processing** If training on image patches, then reshape rows of \( W \) to display convolutional filters.

1. Given the measurements \( y_t \) and current \( W \), the lower-level reconstruction problem is solved iteratively using ADMM to obtain an estimate of \( x^*_t(W) \).
2. The sign vector \( \text{sign}(W x_t(W)) \) is obtained after hard-thresholding \( W x_t(W) \) with a small threshold parameter \( \gamma \). The sign vector is of paramount importance as it decides the row-split of \( W \) into \( W_0 \) and \( W_\pm \). Hence, we run ADMM for enough iterations to obtain empirical convergence of the sign pattern.

3. Then, using (3.4) and (3.5), we obtain the gradient of the upper level cost on a single training pair, i.e., \( \nabla_W Q \) which is the row-concatenation of \( \nabla_{W_0} Q \) and \( \nabla_{W_\pm} Q \). Averaging the gradients over the samples in the batch yields the minibatch gradient.

At the end of each batch, we update the matrix \( W \) based on the learning rate \( \alpha \) and the mini-batch gradient. The updated \( W \) is used in the next batch in step (1) above.

Our method of gradient calculation can also be extended when the training pairs are image patches instead of 1D signals. For image denoising experiments, image patches of size \( \sqrt{n} \times \sqrt{n} \) with an overlap stride of \( r \) were extracted from images. Then the 2D patches are converted to 1D arrays as a pre-processing (first) step. In the end, the rows of the learned \( W \) are reshaped to look like convolutional filter patches. Except this first and last step, all the intermediate steps are the same for both 1D signals and image patches.

5. **Experimental results.** To demonstrate how well BLORC learns the transform matrix \( W \), we perform a series of denoising experiments for 1D signals and also for images from the Urban-100 dataset.

5.1. **Denoising 1D signals.** Here, synthetic 1D signals were generated to be sparse with respect to a specific transform that also provides a baseline to compare our learned transforms with.

In our experiment, we generate \( M = 4000 \) training pairs \( (x_t, y_t) \), where the \( x_t \)'s are piece-wise constant signals of length \( n = 64 \) (with peak value normalized to 1) and \( y_t \)'s are noisy versions with additive i.i.d. Gaussian noise with standard deviation \( \sigma = 0.1 \). Figure 2(a) shows a single pair of such \( (x_t, y_t) \). We perform minibatch gradient descent with batch size \( B = 100 \) and run the algorithm for \( E = 750 \) epochs. The learning rate was chosen to be \( \alpha = 10^{-4} \) and the sign threshold was \( \gamma = 10^{-3} \). The learned transform is shown in Figure 2(b). We repeated the experiment with the same parameters but with \( x_t \)'s chosen as smoothly varying signals of different harmonics that are sparse in the discrete cosine transform (DCT) domain as in Figure 2(d). The learned transform is row-rearranged such that it has maximum correlation with the 1D-DCT matrix and is shown in Figure 2(e). The learned transforms of Figures 2(b) and 2(e) capture the same intuition as the standard finite difference transform and the 1D-DCT transform, respectively, but in addition, they also have slight novel features learned for the denoising task. Experimental results indicate that the learned transforms perform better than the standard transforms for denoising on a test-set. The average PSNR for 20 piecewise-constant test signals denoised using the learned \( W \) was 26.2 dB whereas that using the standard transform was 25.8 dB, with the PSNR of the noisy signals being 19.5 dB. The regularization parameter for the standard transform was chosen using golden-section search so that the reconstruction minimized the upper-level cost.

We also perform a 1D denoising experiment comparing an unsupervised analysis dictionary learning method and our supervised approach BLORC.

For the unsupervised approach, the analysis dictionary \( (W) \) is learned by minimizing the objective \( \sum_{t=1}^T \|W x_t\|_1 \), where \( x_t \) is the \( t^{th} \) clean signal, but enforcing an orthogonal constraint
Bilevel Learning of $\ell_1$ Regularizers

Figure 2: 1D training pairs $(x_t, y_t)$ (left column) and their corresponding learned transform $\tilde{W}$ (middle column). a) TV-sparse training signal and its denoised version using learned transform. The noisy signal was denoised using the learned transform. b) Learned transform for piecewise constant signals. c) The analytical finite difference matrix (TV). d) DCT-sparse training signal and its denoised version using learned transform. e) Learned transform for DCT-sparse signals. f) The analytical 1D DCT matrix

5.2. Image denoising. Extending the BLORC algorithm to image patches, we learn reasonable transforms as well. We first demonstrate a simple experiment to denoise toy images which have directional stripes. This is a simple verification experiment, because we can guess how the sparsifying convolutional filters for such images may look.

We chose images (normalized) of size $256 \times 256$ with directional patterns (vertical stripes
and diagonal stripes) and generated their noisy versions with i.i.d. Gaussian noise with \( \sigma = 0.1 \). We extracted image patches of size \( 8 \times 8 \) with an overlap stride of 7. The reason for choosing a large stride was to ensure that distinct training pairs are obtained. As a preprocessing step, the image patches of size \( 8 \times 8 \) were vectorized to size \( n = 64 \).

All the parameters and the intermediate steps of the minibatch gradient descent were unchanged from the 1D signal experiment above. The learned convolutional filters in Figs. 4(b) and 4(d) look orthogonal to their image counterparts in Figs. 4(a) and 4(c), respectively, which is the expected output. In particular, these transforms are learned not only to sparsify the images but to minimize the gap between reconstructions and the ground truth.

Next, we show simple illustrative image denoising results using the Urban-100 dataset. The study is meant to provide some understanding of the behavior of supervised learning via BLORC and not an exhaustive demonstration of image denoising performance. We compare the results of image denoising by BLORC with the methods BM3D [15] and Analysis KSVD [47]. Both these algorithms are well-known to work on images with blocks, structures and orientations and hence comparing with these denoising algorithms gives

---

Figure 3: Learned transforms in BLORC and with unsupervised analysis dictionary learning and denoising comparisons.

Figure 4: Training images (only clean) with a specific orientation (left) and learned sparsifying filters (right): a) image with vertical stripes (vectorized patches extracted from the image are used as training set), b) learned filters for vertical stripes, c) image with diagonal stripes, and d) learned filters for diagonal stripes.
Figure 5: Patch-based Filters learned from images in the Urban-100 dataset. The filters capture orientations in the images. For each image, we show a subset of the rows of the learned $W$ as patches/filters.

us a standard baseline for the performance of BLORC. We fine-tuned the hyper-parameters of these algorithms to achieve the best denoising PSNR at test time.

For image denoising with BLORC, the images are split into training and validation sets. In Figure 5, we show the three images in the training set used to learn three sets of convolutional filters. We denote these learned transforms by $W_1$, $W_2$, and $W_3$, respectively. The dimension of each of these learned transforms is $n \times n$, with $n$ being the unravelled patch dimension. In the experiments, we chose the square patch dimension to be $8 \times 8$, hence $n = 64$. Note that these filters have been learned at a patch level, i.e., by extracting patches from the training images.

We have purposefully chosen the training set to contain images with different orientations so that we learn convolutional filters capturing different orientations in a supervised manner. To use the learned filters for denoising images in a test set, we formed a row-stacked version of these three transforms which we denote as $W_{\text{learned}} \in \mathbb{R}^{3n \times n}$.

To test the effectiveness of the learned filters for denoising, we use this stacked transform matrix $W_{\text{learned}}$ to denoise a different set of test images, labeled Image-1 to Image-4, respectively as shown in Table 2. We solve the denoising problem at the patch level, where we solve the following optimization problem using ADMM to obtain denoised patches $\hat{x}_j$:

$$\{\hat{x}_j\} = \arg \min_{\{x_j\}} \sum_{j=1}^{n} \left( \|x_j - P_j y\|_2^2 + \|W_{\text{learned}} x_j\|_1 \right)$$

The operator $P_j$ extracts the $j$th patch from the image. The regularizer makes sure that the solution at the patch level is sparse w.r.t. the transform $W_{\text{learned}}$. The denoised image is
obtained after spatially aggregating the denoised patches $\hat{x}_j$.

| Image/method | Clean Image | BM3D | Analysis KSVD | BLORC (Ours) |
|---------------|-------------|------|---------------|--------------|
| **Image-1**   | ![Clean Image](image1.png) | ![BM3D](image1 BM3D.png) | ![Analysis KSVD](image1 Analysis KSVD.png) | ![BLORC (Ours)](image1 BLORC.png) |
| PSNR          | -           | 26.27 | 25.92         | 27.02        |
| **Image-2**   | ![Clean Image](image2.png) | ![BM3D](image2 BM3D.png) | ![Analysis KSVD](image2 Analysis KSVD.png) | ![BLORC (Ours)](image2 BLORC.png) |
| PSNR          | -           | 25.51 | 25.30         | 25.20        |
| **Image-3**   | ![Clean Image](image3.png) | ![BM3D](image3 BM3D.png) | ![Analysis KSVD](image3 Analysis KSVD.png) | ![BLORC (Ours)](image3 BLORC.png) |
| PSNR          | -           | 21.90 | 21.34         | 22.17        |
| **Image-4**   | ![Clean Image](image4.png) | ![BM3D](image4 BM3D.png) | ![Analysis KSVD](image4 Analysis KSVD.png) | ![BLORC (Ours)](image4 BLORC.png) |
| PSNR          | -           | 21.22 | 22.05         | 22.34        |

Table 2: Image denoising for $\sigma = 38.25$.

We conclude from Table 3 that for a lower noise level of $\sigma = 25.5$, the unsupervised analysis dictionary learning approach Analysis K-SVD (that learns a separate dictionary to denoise each noisy image) slightly outperforms BLORC. However, this is not the case for a higher noise level such as $\sigma = 38.25$. The supervised approach BLORC performs slightly better in terms of the PSNR metric for a higher level of noise.

Unsupervised dictionary learning methods like Analysis K-SVD rely on the structure of (unpaired) corrupted signals/images to learn the dictionary atoms. These structures of the signals can get buried when learning from signals with significant levels of noise. On the other hand, BLORC utilizes both the corrupted signal and the ground truth signals to learn the transform (only once on training set). Moreover, the transforms are learned to minimize image quality metrics of interest. Hence, BLORC can learn relevant features even in the presence of significant amounts of noise.

5.3. **Comparison with other differentiable solvers.** While the BLORC algorithm uses an explicit form of the gradient, in automatic differentiation approaches, the task is divided into
BILEVEL LEARNING OF $\ell_1$ REGULARIZERS

|                | Noisy case | BM3D   | Analysis K-SVD | BLORC (ours) |
|----------------|------------|--------|----------------|--------------|
| $\sigma = 38.25$ |            |        |                |              |
| PSNR           | 8.24       | 23.72  | 23.65          | 24.16        |
| $\sigma = 25.50$ |            |        |                |              |
| PSNR           | 10.4       | 26.83  | 26.60          | 26.17        |

Table 3: Performance of different methods for image denoising (averaged over 4 images) using the PSNR (in dB) metric for two different noise levels.

A sequence of differentiable operations as computational graphs on which backpropagation is performed through the chain rule. This division into a sequence of operations and calculating gradients for each node of the graph can take significant time, which can be bypassed if an explicit form of the gradient already exists that connects both the upper level and the lower level problems. This advantage in time can be crucial for larger datasets and batch sizes. As a demonstration of this, we calculate $\nabla WQ$ for a training pair $(x_t, y_t)$ using three methods with the $W = I$ initialization. We note the time and accuracy for calculating a single instance of $\nabla WQ$ and average them out over 100 such different training pairs in Table 4. In the first method, we use the direct expressions in (3.4) and (3.5) to get $\nabla WQ$, which we denote as "BLORC" in Table 4. For the second method, we used the CVXPY package to run an iteration of optimization over the bilevel problem for the same training pair and obtained the gradient. Finally, as the third method, we used PyTorch to calculate the gradient of our closed-form expression in (3.1).

| Gradient Method | $n = 36$ | $n = 64$ |
|-----------------|----------|----------|
|                 | Time (ms) | Error    | Time (ms) | Error    |
| BLORC (ours)   | 7.54     | 3.2e-09  | 12.65     | 5.13e-09 |
| PyTorch        | 8.75     | 3.7e-09  | 14.03     | 5.35e-09 |
| CVXPY          | 17.85    | 4.8e-05  | 41.86     | 3.2e-05  |

Table 4: Time and error comparisons for gradient calculation averaged over 100 different single training pairs $(x_t, y_t)$.

As the baseline for our comparisons, we calculated the numerical gradient of the upper level problem by noting the incremental change in the cost for incremental changes in each element of the matrix $W$. The errors for the three methods in Table 4 have been calculated with the ground truth value being set to the one from the numerical gradient method. The analytical form of the gradient in BLORC makes it faster and more accurate compared to automatic differentiation approaches as is evident from Table 4. It is noteworthy that even when the PyTorch method uses the closed-form expression, BLORC marginally outperforms
5.4. Comparison with smooth differentiable penalty. We perform an experiment to compare the transform when learned with a) an $\ell_1$ norm functional and with b) an element-wise Huber-loss function which is given as:

$$L_\delta(x) = \begin{cases} \frac{1}{2}x^2 & \text{for } |x| < \delta \\ \delta(|x| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

The quadratic approximation of the Huber loss function near zero makes it less effective in promoting sparsity than the classical $\ell_1$ norm. We perform an experiment to learn the transform using a Huber-loss penalty using a bilevel framework as follows:

$$\arg \min_W Q(W) = \frac{1}{2}\|x^*_t(W, y_t) - x_t\|^2_2$$

$$\text{s.t. } x^*_t(W, y_t) = \arg \min_x \frac{1}{2}\|x - y_t\|^2_2 + \sum_{i=1}^m L_\delta((W x)_i)$$

We use the autograd in CVXPY to obtain the gradient through the lower-level problem to obtain $\nabla_W Q(W)$. We perform gradient descent until convergence. The comparison between the learned transforms (note that for BLORC, we still use our direct closed-form expressions) and their denoising abilities is shown in Figure 6.

Let the transform learned using the $\ell_1$ loss be denoted as $W_{\ell_1}$, and the transform learned using the Huber-loss functional be denoted as $W_{Hub}$. The Huber-parameter $\delta$ was optimized on a test dataset over a range of values in $[\|y\|_\infty^4, \frac{3}{4}\|y\|_\infty^4]$ and the setting achieving the best test denoising metric (PSNR) was chosen. A careful optimization of $\delta$ is required because a higher value of $\delta$ would cause the regularizer to act more like a smooth Tikhonov regularizer and a low value of $\delta$ would make the penalty weak.

We observe from our experiments that $W_{Hub}$ (learned with careful choice of $\delta$) does not sharply promote sparsity for piecewise constant signals. This is quite evident when each row of $W_{Hub}$ and $W_{\ell_1}$ are plotted together. As in Figure 6, when $W_{Hub}$ is used to denoise a noisy piecewise-constant signal, it fails to reproduce the sharp edges unlike $W_{\ell_1}$.

6. Identifiability of transform under various noise levels. It is of interest to inspect the performance of the BLORC algorithm under various noise-levels. Also, therein lies the question: will the algorithm converge to the same the minimizer/transform for increasing noise-levels? To answer this question, we first perform several experiments to learn the transform matrix $W$ with piecewise constant training signal pairs $(x_t, y_t)_{t=1}^T$ for different noise-levels. Each $x_t$ is a piece-wise constant signal and the noisy measurement is generated as $y_t = x_t + \epsilon$, where $\epsilon \in \mathcal{N}(0, \sigma^2 I)$. We chose 8 noise-levels as $\sigma = 0, 0.02, 0.05, 0.07, 0.1, 0.2, 0.5, \text{and } 0.7$ (all $x_t$’s values are from 0 to 1) and performed the BLORC experiment to obtain the learned transform $W$ for each of the noise levels. The learned transforms for the different noise levels are shown in Figure 7. For all the 8 experiments, the transform was initialized to be $W = I$, and we see in Figure 7 that except for $\sigma = 0.5$ and 0.7, $W$ converged to something very close to the finite-differences matrix (lets call it $W^*$).
Now consider the noiseless case ($\sigma = 0$), for which we have $x_t = y_t$. From Figure 7, we observe that the finite-difference matrix ($W^*$) is a minimizer for the upper-level cost $Q(W)$. It is also easy to observe that $W = 0$, the zero-matrix is a minimizer. In both these realizations of $W$, the upper level loss ($Q(W)$) reaches its minimum value, i.e., zero.

The problem gets interesting when the noise level (standard deviation) is increased beyond $\sigma = 0$. Note that for $W = 0$, the upper level loss is roughly $\sigma^2$, i.e., linear in the noise variance. This is obvious since the lower level reconstruction is just $y_t$, which makes the upper level loss $Q(0) = \frac{1}{T} \sum_{t=1}^{T} \|x_t - y_t\|_2^2 \approx \sigma^2$. Thus, with higher noise levels, the zero matrix would not be expected to remain a local minimizer anymore as the upper level loss increases drastically with increase in $\sigma$. However, we could expect $W^*$ to be a minimizer even with noise.

It is noteworthy that beyond noise level $\sigma = 0.5$, the pattern of the learned transform $W$ breaks. This is because beyond noise-level $\sigma = 0.5$, piecewise-constant structure in the corrupted signals are not preserved.

Another interesting observation in Figure 7 is that with increasing noise level, the scale of the learned transform increases (the colorbar range denotes the scale). This confirms the rather common occurrence in denoising problems. With increasing noise-level, reconstructing from the noisy signal becomes increasingly difficult, hence the regularization term is learned to take more weight than the data-fidelity term to achieve a more stable reconstruction in the lower level problem. Even at significant noise levels like $\sigma = 0.2$ (which is 20% of the peak intensity...
Figure 7: BLORC learned $W$’s for various noise levels. In all experiments, BLORC was initialized with $W = I$. Supervised $(x_t, y_t)$ pairs used in the training set were piecewise-constant signals.

Figure 8: Training loss and validation loss vs. number of epochs for training with various noise levels. Let $W_i$ denote the learned transform at epoch $i$. Training loss is the upper-level loss $Q(W_i) = \frac{1}{T} \sum_{t=1}^{T} \| x_t^*(W_i, y_t) - x_t \|_2^2$. The validation set $(x_l, y_l)_{l=1}^L$ have been generated independently from the training set but with the same noise-level. The validation loss is given as $\frac{1}{L} \sum_{l=1}^{L} \| x_l^*(W_i, y_l) - x_l \|_2^2$. Here $x_l^*(W_i, y_l)$ is the result of denoising $y_l$ with learned transform at $i$th epoch $W_i$.

$\sigma = 0.5$ $\sigma = 0.1$ $\sigma = 0.02$

Finally, in Figure 8, the upper-level loss $Q(W)$ is plotted for training and validation signals across training epochs. The validation pairs consisting of ground truth signals and noisy measurements were generated using the same noise levels as that of the training set that was used to learn the transform. For noise levels $\sigma = 0.02$ and $0.1$, the learned transform $W$
generalizes well and has a good denoising performance on unseen validation data. This is observed as the validation loss for $\sigma = 0.02$ and $\sigma = 0.1$ decreases to a significantly low value. However, for $\sigma = 0.5$, the validation loss does not decrease significantly and hence denoising using the learned transform for $\sigma = 0.5$ yields poor performance on the unseen validation set. This is also justified from the fact that in Figure 8, the learned transform for $\sigma = 0.5$ does not resemble the structures in a finite-difference matrix.

7. Proofs and analysis of closed-form and gradient expressions. In this section, we present the derivations of the closed-form expressions and gradients and an analysis of the closed-form expression on Stiefel manifolds. First, we present a proof for Theorem 3.1.

7.1. Obtaining the closed-form: Proof for Theorem 3.1. We propose to solve signal reconstruction problems using a sparsifying analysis operator that is learned from training data in a supervised way. Our learning problem is

\begin{equation}
\arg \min_{W} Q(W), \text{ where } Q(W) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{2} \|x^*_t(W, y_t) - x_t\|_2^2,
\end{equation}

where $x_t$ and $y_t \in \mathbb{R}^n$ are the $t$th training signal and its corresponding noise-corrupted version, $W \in \mathbb{R}^{n \times n}$ is a sparsifying operator we intend to learn with the help of the corresponding lower level problem,

\begin{equation}
x^*_t(W, y_t) = \arg \min_{x} \frac{1}{2} \|x - y_t\|_2^2 + \beta \|Wx\|_1.
\end{equation}

Let $c(W)$ denote the sign pattern associated with a given $W$, i.e., $c(W) = \text{sign}(Wx^*(W))$, where $\text{sign}(z)_i$ is defined to be -1 when $|z| < 0$; 0 when $|z| = 0$; and 1 when $|z| > 0$. Note that we omit the $t$-dependence from $x^*_t(W)$ as it is not relevant for the derivation. Considering a fixed sign-pattern $c_0 = c(W)$, we define matrices that pull out the rows of $W$ that give rise to zero, negative, and positive values in $Wx^*(W)$. Let $k_0$, $k_{\neq 0}$, $k_<$, and $k_>$ denote the number of zero, nonzero, negative, and positive elements of $c$, respectively. Similarly, let $[\pi_0]_m$, $[\pi_{<0}]_m$, and $[\pi_{>0}]_m$ denote the indices of the $m$th zero, negative, and positive element of $c_0$, respectively. Let $S_0 \in \mathbb{R}^{k_0 \times k}$ and $S_{\pm} \in \mathbb{R}^{k_{\neq 0} \times k}$ be defined as

\begin{equation}
[S_0]_{m,n} = \begin{cases} 1 & \text{if } [\pi_0]_m = n; \\ 0 & \text{otherwise;} \end{cases} \quad \text{and} \quad [S_{\pm}]_{m,n} = \begin{cases} 1 & \text{if } [\pi_{>0}]_m = n; \\ -1 & \text{if } [\pi_{<0}]_{m-k_0} = n; \\ 0 & \text{otherwise.} \end{cases}
\end{equation}

The nonzero sign pattern is $s = S_{\pm}c_0$. Let $W_0 = S_0W$ and $W_{\pm} = S_{\pm}W$ contain the rows of $W$, whose indices are given by the sets $k_0$ and $k_{\neq 0}$, respectively. With these notations in place, we can write

\begin{equation}
x^*(W) = \arg \min_{x} \frac{1}{2} \|x - y\|_2^2 + \beta \|Wx\|_1, \quad \text{s.t.} \quad W_0x = 0
\end{equation}
for all $W$ such that $c(W) = c_0$. This is true because whenever $c(W) = c_0$, the minimizer of (3.1) is feasible for (7.3). Similarly, we use $W_{\pm}$ to simplify the $\ell_1$ norm term,

$$
(7.4) \quad x^*(W) = \arg\min_x \frac{1}{2}\|x - y\|_2^2 + \beta s^T W_{\pm} x, \quad \text{s.t.} \quad W_0 x = 0,
$$

which holds again for all $W$ such that $c(W) = c_0$.

Now that we have transformed the problem into an equality-constrained quadratic minimization, we can use standard results (e.g., see [8], Section 10.1.1) to state the KKT conditions for (7.4) as

$$
(7.5) \quad \begin{bmatrix} I & W_0^T \\ W_0 & 0 \end{bmatrix} \begin{bmatrix} \nu^* \\ \nu \end{bmatrix} = \begin{bmatrix} y - (\beta s^T W_{\pm})^T \\ 0 \end{bmatrix},
$$

where the underbraces give names ($A$, $z$, and $b$) to each quantity to simplify the subsequent notation. Because $I$ is nonsingular, $A$ is invertible whenever $W_0$ has full row rank [8]. Lastly, in order to extract the part corresponding to $x^*(W)$, we define a selection matrix $P_x = [I \ 0]$. The part corresponding to $x^*(W)$ can be extracted as $x^*(W) = P_x A^{-1} b$. If $W_0$ has full row rank, then by block matrix inversion formula corresponding to the first term we have the final closed-form expression:

**Lemma 7.1.** Given a block matrix partitioned into four blocks, it can be inverted blockwise in the following manner: If $P = \begin{bmatrix} L & M \\ N & O \end{bmatrix}$ where $L$ and $O$ are arbitrary sized, and $M$ and $N$ are conformable for partitioning, then $P^{-1} = \begin{bmatrix} L + L^{-1}M(O - NL^{-1}M)^{-1}NL^{-1} & -L^{-1}M(O - NL^{-1}M)^{-1}M \\ -M^T(L^{-1}M)^{-1}NL^{-1} & (O - NL^{-1}M)^{-1} \end{bmatrix}$ [5]. Furthermore, $L$ and its schur complement $(O - NL^{-1}M)$ must be invertible.

Now, according to Lemma 7.1 and considering the block matrix $A = \begin{bmatrix} I & W_0^T \\ W_0 & 0 \end{bmatrix}$, the condition for $A$ to be invertible here is for $-W_0 W_0^T$ (Schur complement of $A$) to be invertible. The necessary and sufficient condition for $W_0 W_0^T$ to be invertible is for $W_0$ to have a full row-rank. Now, extracting the term corresponding to $x^*(W)$ from $z$ using the block matrix inversion formula in Lemma 7.1, we would have:

$$
x^*(W) = (I + W_0 W_0^T (0 - W_0 I^{-1} W_0^T)^{-1} W_0 I^{-1} ) (y - \beta W_{\pm}^T s) = (I - W_0 W_0^T)^{-1} W_0 (y - \beta W_{\pm}^T s) = P_N(W_0)(y - \beta W_{\pm}^T s).
$$

Here, $P_N(W_0)$ denotes the projection operator onto the null-space of $W_0$.

We also observe that this similar closed-form expression has also been obtained in Equation (33) of [52] by formulating the dual problem.
Also, it is noteworthy that the closed-form expression derived above holds even when $W_0$ is not full row-rank but with a slight modification as follows. Note that when $W_0$ is not full row-rank, $A$ defined in (7.5) is also rank-deficient. Hence to calculate the closed-form $x^*(W)$, we use the expression $x^*(W) = P_x A^T b$, where $A^\dagger$ denotes the pseudo-inverse of $A$. It is then, straightforward to prove that the updated closed-form expression is given as $x^*(W) = (I - W_0^T(W_0 W_0^T) \dagger W_0)(y - \beta W_\pm^T s)$, where $\dagger$ denotes the pseudo-inverse operator. This can be derived from the formula of psuedo-inverse of block-matrices as given in [24]. The term $(W_0 W_0^T) \dagger$ becomes $(W_0 W_0^T)^{-1}$ only when $W_0$ is full row-rank which is exactly the boxed expression above.

Next, we present the derivations of the gradient of the closed-form expression that we stated in Corollary 3.3.

### 7.2. Gradients of the closed-form: Proof for Corollary 3.3

Now that we have got a closed-form expression for the lower level cost, we can rewrite the original bilevel optimization problem (1.2) as

$$
(7.6) \quad \text{arg min}_W Q(W), \quad \text{where} \quad Q(W) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{2} \|x^*(W, y_t) - x_t\|_2^2,
$$

$$
(7.7) \quad \text{s.t.} \quad x^*(W, y_t) = (I - P_{row(W_0)})(y_t - \beta W_\pm^T s)
$$

Since we perform SGD based updates, we calculate the gradient of the upper level cost function for each training pair and take a gradient step. Let $Q(W) = \frac{1}{T} \|x^*(W, y_t) - x_t\|_2^2$ be the upper level cost function for each training pair, then $\partial Q = \nabla_{x^*} Q^T \partial x^*$.

Next, we aim to differentiate (7.7) with respect to the elements of $W$. We proceed by differentiating separately with respect to $W_\pm$ and $W_0$ because they form a partition of the rows of $W$. The $W_\pm$ part follows from simple matrix calculus rules [37] as

$$
(7.8) \quad \partial x^* = -\beta P_N(W_0) \partial W_\pm^T s.
$$

The $W_0$ part requires differentiation through a null space projection, for which we rely on Theorem 4.3 in [23]. The result is

$$
(7.9) \quad \partial x^* = -(W_0^\dagger \partial W_0 P_{N(W_0)} + (W_0^\dagger \partial W_0 P_{N(W_0)})^T)(y - \beta W_\pm^T s),
$$

Finally, we are interested in computing the gradient with respect to $W$ of an upper-level cost functional, $Q : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$; this can be achieved via the chain rule (writing $\partial Q = \nabla_{x^*} Q^T \partial x^*$, substituting $\partial x^*$, and rearranging into canonical form [37]), which results in

$$
(7.10) \quad \nabla_{W_\pm} Q = -\beta s \nabla_{x^*} Q^T P_N(W_0)
$$

$$
(7.11) \quad \nabla_{W_0} Q = - (P_{N(W_0)}(q \nabla_{x^*} Q^T + \nabla_{x^*} Q q^T)W_0^+)^T
$$

with $q = y - \beta W_\pm^T s$. Here, we choose $Q(W) = \frac{1}{T} \|x^*(W) - x_t\|_2^2$, so $\nabla_{x^*} Q = \frac{1}{T}(x^* - x_t)$. 


Due to the orthogonality condition on the rows, we have the inner-product of a transform row with the measurement vector is either of the closed-form expression projected to the Stiefel manifold using smooth retraction [6]. The non-differentiable points (due to the orthogonal constraint) directly have the orthogonal constraint.

Next, replacing \( \partial x^* \) with Equation (7.9), we can similarly have a relation between \( \partial Q \) and \( \partial W_0 \).

\[
\partial Q = \nabla_{x^*} Q^T \partial x^* = -\beta \text{Tr}(\nabla_{x^*} Q^T P_{N(W_0)} \partial W_0^T s)
\]

\[
= -\beta \text{Tr}(s \nabla_{x^*} Q^T P_{N(W_0)} \partial W_0^T)
\]

Recalling that in Section 3.3, if the differential of a scalar and matrix is related as \( \partial Q = \text{Tr}(E^T \partial W_0) \), then \( \nabla W_0 = E \). Hence, we have from the last line, \( \nabla W_0 = -\beta(s \nabla_{x^*} Q^T P_{N(W_0)}) \).

Next, replacing \( \partial x^* \) with (7.9), we can similarly have a relation between \( \partial Q \) and \( \partial W_0 \).

We see now that the differentials are related by the form \( \partial Q = \text{Tr}(E^T \partial W_0) \), hence we directly have \( \nabla W_0 = -\beta(P_{N(W_0)}(s \nabla_{x^*} Q^T + \nabla_{x^*} Q q^T) W_0^+ \partial W_0) \).

7.3. Analysis of closed-form expression on the Stiefel manifold. This section analyses the closed-form expression for the denoising problem obtained on the Stiefel manifold (under the orthogonal constraint) \( W W^T = I \). In this case, gradients from the upper-level can be projected to the Stiefel manifold using smooth retraction [6]. The non-differentiable points of the closed-form expression \( x^*_i(W) = W^T S_\lambda(W y_i) \) are given by \( |W y_i| = \pm \lambda \), i.e., an inner-product of a transform row with the measurement vector is either \( \lambda \) or \(-\lambda \). We show that these non-differentiable points form a set of measure zero on the Stiefel manifold.

For simplicity, we start by considering a matrix in \( \mathbb{R}^{2 \times 2} \), which also lies on the Stiefel manifold. Such a matrix would take the form \( W = \begin{bmatrix} w_1 & \sqrt{1-w_1^2} \\ w_2 & \sqrt{1-w_2^2} \end{bmatrix} \), as the rows are normalized.

Due to the orthogonality condition on the rows, we have \( w_1 w_2 + \sqrt{(1-w_1^2)(1-w_2^2)} = 0 \). The locus of the curve forms the Stiefel manifold in the domain of \( [w_1, w_2] \in \mathbb{R}^2 \), and is shown by the green circular curve in Figure 9. The non-differentiable points lie on \( |W y_i| = \pm \lambda \) (for a generic \( y \)). Without loss of generality, we assume the point of non-differentiability holds only on the
first index of the vector, i.e., $[Wy]_1 = \pm \lambda$. It is easy to see that in $\mathbb{R}^2$, the non-differentiability points of the closed-form expression would lie on the curve $w_1 + b\sqrt{1 - w_1^2} = c$, as denoted by the black dashed lines in Figure 9. These lines intersect the manifold curve (shown in green) only at finite points as shown by the red blobs. The dimension of the Stiefel manifold for matrices in $\mathbb{R}^{2\times2}$ is $5$, whereas the dimension of the non-differentiable points on the manifold (here red points) is $0$. Then, according to Theorem 2.22 in [48], the non-differentiable points form a set of Lebesgue measure $0$ as this intersection has a dimension lower than the dimension of the manifold.

Next, we provide a brief discussion for matrices in $\mathbb{R}^{n\times n}$ lying on the Stiefel manifold ($V_n(\mathbb{R}^n)$), or the set of orthogonal $n$-frames in $\mathbb{R}^n$. A non-constrained matrix in $\mathbb{R}^{n\times n}$ has dimension of $n^2$ as there are $n^2$ free parameters. Now as each row is normalized, the parameters in each row are constrained by the equation $\sum_{j=1}^n w_{ij}^2 = 1$. Hence, for $n$ rows, the degrees of freedom decrease by $n$. Moreover, as the rows are orthogonal to each other, each (independent) orthogonality constraint is captured by $\sum_{j=1}^n w_{ij}w_{kj} = 0$ for $k \neq i$. As there are $\binom{n}{2}$ distinct pairs of rows, the degrees of freedom reduce by $\binom{n}{2}$ from the orthogonality conditions. Hence, the effective dimension of a Stiefel manifold for matrices in $\mathbb{R}^{n\times n}$ could be seen as $n^2 - n - \binom{n}{2} = \frac{n(n-1)}{2}$.

Now without loss of generality, consider that the closed-form expression is not differentiable only in the first index, i.e., $[Wy]_1 = \lambda$ (could also be $-\lambda$ equivalently). So the variables in the first row of $W$ are constrained by an extra equation. This decreases the overall degrees of freedom by one and hence the effective dimension of the intersection of the hyperplane $[Wy]_1 = \lambda$ ($\lambda > 0$) and the Stiefel manifold is $\frac{n(n-1)}{2} - 1$, which is one less than the dimension of the manifold itself. Thus, these non-differentiable points would form a set of measure zero on the Stiefel manifold.

8. Conclusions. This paper presented an approach for supervised learning of sparsity-promoting nonsmooth ($\ell_1$) regularizers for denoising problems. The underlying training problem is a challenging bilevel optimization problem, where the upper level loss is a task-based one (e.g., mean squared error for denoising) and the lower level problem is a variational problem for denoising, whose solution is used in the upper level loss. Our approach, BLORC, allows learning the sparsifying operator.

---

5 The green curve can be parameterized by $[\sin(\theta), \cos(\theta)]$ and hence has only a single degree of freedom, i.e., on $\theta$, where $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}] \cup [\frac{3\pi}{2}, 2\pi]$. The angle restriction is to maintain row-orthogonality.

6 If the non-differentiability holds in other indices as well, that does not increase the dimension of the set in which these points lie. Hence, the argument still holds.
in the lower level problem by exploiting local closed-form expressions for the solution of the variational problem. The closed-form expressions enable computing gradients directly and efficiently with respect to the operator parameters. Experimental results show the ability of BLORC to learn underlying sparsifying operators robustly in the presence of noise for both 1D signals and 2D images. For images, the proposed approach outperforms recent denoising schemes, including unsupervised analysis dictionary learning. In future work, we plan to extend the approach to handle non-unique lower-level minimizers and extend to other inverse problems with complex-valued images such as in magnetic resonance imaging.

Acknowledgments. We thank Jeffrey Fessler and Caroline Crockett, University of Michigan, for helpful discussions and their comments on this work.

REFERENCES

[1] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. Moursi, Differentiating through a cone program, Journal of Applied and Numerical Optimization, 1 (2019), pp. 107–115.
[2] M. Aharon, M. Elad, and A. Bruckstein, K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation, IEEE Transactions on signal processing, 54 (2006), pp. 4311–4322.
[3] K. Akiyama, A. Alberdi, W. Aref, K. Asada, R. Azulay, A.-K. Baczko, D. Ball, M. Baloković, J. Barrett, D. Bentley, et al., First m87 event horizon telescope results. iv. imaging the central supermassive black hole, The Astrophysical Journal Letters, 875 (2019), p. L4.
[4] A. Alessio, P. Kinahan, et al., PET image reconstruction, Nuclear medicine, 1 (2006), pp. 1–22.
[5] D. S. Bernstein, Matrix mathematics, in Matrix Mathematics, Princeton university press, 2009.
[6] P. Birtea, I. Cașu, and D. Comănescu, First order optimality conditions and steepest descent algorithm on orthogonal stiefel manifolds, Optimization Letters, 13, pp. 1773–1791.
[7] C. H. Bischof, A. Carle, P. M. Khademi, and G. Pusch, Automatic differentiation: obtaining fast and reliable derivatives—fast, in Control Problems in Industry, Springer, 1995, pp. 1–16.
[8] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[9] G. T. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium, SIAM Journal on Imaging Sciences, 11 (2018), pp. 2001–2020.
[10] J. B. Campbell and R. H. Wynne, Introduction to remote sensing, Guilford Press, 2011.
[11] S. H. Chan, X. Wang, and O. A. Elgendy, Plug-and-play ADMM for image restoration: Fixed-point convergence and applications, IEEE Transactions on Computational Imaging, 3 (2016), pp. 84–98.
[12] Y. Chen, T. Pock, and H. Bischof, Learning ℓ1-based analysis and synthesis sparsity priors using bi-level optimization, arXiv:1401.4105 [cs.CV], (2014), https://arxiv.org/abs/1401.4105.
[13] Y. Chen, R. Ranftl, and T. Pock, Insights into analysis operator learning: From patch-based sparse models to higher order MRFs, IEEE Transactions on Image Processing, 23 (2014), pp. 1060–1072.
[14] C. Crockett and J. A. Fessler, Bilevel methods for image reconstruction, arXiv preprint arXiv:2109.09610, (2021).
[15] K. Darov, A. Foi, V. Katkovnik, and K. Egiazarian, Image denoising with block-matching and 3d filtering, in Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning, vol. 6064, International Society for Optics and Photonics, 2006, p. 606414.
[16] J. C. De los Reyes, C.-B. Schönlieb, and T. Valkonen, Bilevel parameter learning for higher-order total variation regularisation models, Journal of Mathematical Imaging and Vision, 57 (2017), pp. 1–25.
[17] D. Donoho, Compressed sensing, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306.
[18] I. A. Elbakri and J. A. Fessler, Statistical image reconstruction for polyenergetic X-ray computed tomography, IEEE transactions on medical imaging, 21 (2002), pp. 89–99.
[19] J. A. Fessler, Optimization methods for magnetic resonance image reconstruction: Key models and optimization algorithms, IEEE signal processing magazine, 37 (2020), pp. 33–40.
[20] S. Fiege, A. Walther, K. Kulshreshtha, and A. Griewank, Algorithmic differentiation for piecewise
smooth functions: a case study for robust optimization, Optimization Methods and Software, 33 (2018), pp. 1073–1088.

[21] A. Ghosh, M. T. McCann, and S. Ravishankar, Bilevel Learning of $\ell_1$ Regularizers with Closed-Form Gradients (BLORC), in ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 1491–1495.

[22] D. Gilbert, G. Ongie, and R. Willett, Deep equilibrium architectures for inverse problems in imaging, IEEE Transactions on Computational Imaging, 7 (2021), pp. 1123–1133.

[23] G. H. Golub and V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, SIAM Journal on Numerical Analysis, 10 (1973), pp. 413–432.

[24] C.-h. Hung and T. L. Markham, The moore-penrose inverse of a partitioned matrix $m:=(adbc)$, Linear Algebra and its Applications, 11 (1975), pp. 73–86.

[25] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, Deep convolutional neural network for inverse problems in imaging, IEEE Transactions on Image Processing, 26 (2017), pp. 4509–4522.

[26] D. Kim, S. Ramani, and J. A. Fessler, Combining ordered subsets and momentum for accelerated X-ray CT image reconstruction, IEEE transactions on medical imaging, 34 (2014), pp. 167–178.

[27] V. Kouni, G. Paraskevopoulos, H. Rauhut, and G. C. Alexandropoulos, Admm-dad net: A deep unfolding network for analysis compressed sensing, in ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 1506–1510.

[28] K. Kunisch and T. Pock, A bilevel optimization approach for parameter learning in variational models, SIAM Journal on Imaging Sciences, 6 (2013), pp. 938–983.

[29] D. J. Larkman and R. G. Nunes, Parallel magnetic resonance imaging, Physics in Medicine & Biology, 52 (2007), p. R15.

[30] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, Object recognition with gradient-based learning, in Shape, contour and grouping in computer vision, Springer, 1999, pp. 319–345.

[31] W. Lee, H. Yu, X. Rival, and H. Yang, On correctness of automatic differentiation for non-differentiable functions, Advances in Neural Information Processing Systems, 33 (2020), pp. 6719–6730.

[32] A. Levis, D. Lee, J. A. Tropp, C. F. Gammie, and K. L. Bouman, Inference of black hole fluid-dynamics from sparse interferometric measurements, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2340–2349.

[33] 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, 34 (2021).

[34] M. Lyu and A. Plouss, Filtering in SPECT image reconstruction, International Journal of Biomedical Imaging, 2011 (2011).

[35] J. Mairal, F. Bach, and J. Ponce, Task-driven dictionary learning, IEEE Transactions on Pattern Analysis and Machine Intelligence, 34 (2012), pp. 791–804.

[36] C. C. Margossian, A review of automatic differentiation and its efficient implementation, Wiley interdisciplinary reviews: data mining and knowledge discovery, 9 (2019), p. e1305.

[37] T. P. Minka, Old and new matrix algebra useful for statistics, tech. report, MIT Media Lab, 2000.

[38] V. Monga, Y. Li, and Y. C. Eldar, Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing, IEEE Signal Processing Magazine, 38 (2021), pp. 18–44.

[39] P. Ochs, R. Ranftl, T. Brox, and T. Pock, Bilevel optimization with nonsmooth lower level problems, in Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 654–665.

[40] G. Peyrè and J. M. Fadili, Learning analysis sparsity priors, in Sampling Theory and Applications, Singapore, Singapore, May 2011, p. 4.

[41] S. Ravishankar and Y. Bresler, MR image reconstruction from highly undersampled k-space data by dictionary learning, IEEE transactions on medical imaging, 30 (2010), pp. 1028–1041.

[42] S. Ravishankar and Y. Bresler, Learning sparsifying transforms, IEEE Transactions on Signal Processing, 61 (2012), pp. 1072–1086.

[43] S. Ravishankar and Y. Bresler, Data-driven learning of a union of sparsifying transforms model for blind compressed sensing, IEEE Transactions on Computational Imaging, 2 (2016), pp. 294–309.

[44] S. Ravishankar, J. C. Ye, and J. A. Fessler, Image reconstruction: From sparsity to data-adaptive methods and machine learning, Proceedings of the IEEE, 108 (2020), pp. 86–109.

[45] A. J. Reader and H. Zaidi, Advances in PET image reconstruction, PET clinics, 2 (2007), pp. 173–190.

[46] Y. Romano, M. Elad, and P. Milanfar, The little engine that could: Regularization by denoising
R. Rubinstein, T. Peleg, and M. Elad, *Analysis k-SVD: A dictionary-learning algorithm for the analysis sparse model*, IEEE Transactions on Signal Processing, 61 (2013), pp. 661–677.

W. Rudin, *Real and Complex Analysis P. 2*, McGraw-Hill, 1970.

P. H. Sönnemann, *A generalized solution of the orthogonal procrustes problem*, Psychometrika, 31 (1966), pp. 1–10.

R. A. Schowengerdt, *Remote sensing: models and methods for image processing*, Elsevier, 2006.

P. Sprechmann, R. Litman, T. Ben Yakar, A. M. Bronstein, and G. Sapiro, *Supervised sparse analysis and synthesis operators*, in Advances in Neural Information Processing Systems 26, 2013, pp. 908–916.

R. J. Tibshirani and J. Taylor, *The solution path of the generalized lasso*, The annals of statistics, 39 (2011), pp. 1335–1371.

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, 2013, pp. 945–948.

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.

M. T. Vlaardingerbroek and J. A. Boer, *Magnetic resonance imaging: theory and practice*, Springer Science & Business Media, 2013.

Y. Yang, J. Sun, H. Li, and Z. Xu, *Deep admm-net for compressive sensing mri*, in Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16, Red Hook, NY, USA, 2016, Curran Associates Inc., p. 10–18.

C. Yunjin, P. Thomas, and H. Bischof, *Learning l1-based analysis and synthesis sparsity priors using bilevel optimization*, in NIPS workshop, 2012.

J. Zhang and B. Ghanem, *Ista-net: Interpretable optimization-inspired deep network for image compressive sensing*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 1828–1837.