BILEVEL LEARNING OF ℓ_1 REGULARIZERS WITH CLOSED-FORM GRADIENTS (BLORC)

Avrajit Ghosh[1]*, Michael T. Mccann*[2]*, Saiprasad Ravishankar*[1,3]

[1]Dept. of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI.
[2]Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM.
[3]Dept. of Biomedical Engineering, Michigan State University, East Lansing, MI.

ABSTRACT

We present a method for supervised learning of sparsity-promoting regularizers, which are a key ingredient in many modern signal reconstruction approaches. The parameters of the regularizer are learned to minimize the mean squared error of reconstruction on a training set of ground truth signal and measurement pairs. Training involves solving a challenging bilevel optimization problem with a nonsmooth lower-level objective. We derive an expression for the gradient of the training loss using the implicit closed-form solution of the lower-level variational problem given by its dual problem, and provide an accompanying gradient descent algorithm (dubbed BLORC) to minimize the loss. Our experiments on simple natural images and for denoising 1D signals show that the proposed method can learn meaningful operators and the analytical gradients are calculated faster than standard automatic differentiation methods. While the approach we present is applied to denoising, we believe that it could be adapted to a wide variety of inverse problems with linear measurement models, thus giving it applicability in a wide range of scenarios.

Index Terms— Bilevel optimization, Sparsity-promoting regularizers, Learned Transforms, Gradient descent.

1. INTRODUCTION

Regularized image reconstruction has been used in many applications in imaging and signal processing. Typically, we are interested in solving the optimization problem of the form:

$$\hat{\boldsymbol{x}} = \arg\min_{x} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \mathcal{R}(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is a linear forward model or measurement matrix, $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}$ is a regularization functional, $\boldsymbol{x} \in \mathbb{R}^n$ denotes the underlying signal that we aim to recover, and $\boldsymbol{y} \in \mathbb{R}^m$ denotes its measurements. A prominent theme in designing the regularization functional, $\mathcal{R}$, has been that of sparsity [1]: the idea that the reconstructed signal $\boldsymbol{x}$ should admit a sparse (having a small number of nonzero elements) representation in some domain. While the ℓ_0 "norm" is a canonical measure of sparsity, due to its non-convexity, its relaxed version, the ℓ_1 norm is often used in practice. A common and successful approach to promoting sparsity in signal reconstructions is to use $\mathcal{R}(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{x}\|_1$, where $\boldsymbol{W} \in \mathbb{R}^{k\times n}$ is an analysis dictionary or sparsifying transform. While there are several choices for $\boldsymbol{W}$ that work well in practice (e.g., wavelets, finite differences, discrete cosine transform (DCT)), several authors have also sought to learn $\boldsymbol{W}$ from data – an approach called dictionary (or transform) learning [2, 3]. Methods such as Analysis KSVD [4] and other [5, 6] have been proposed for learning analysis dictionaries from data. Another related class of methods are dubbed sparsifying transform learning [2, 3]. These methods typically learn the operator from unpaired training data (e.g., either clean or noisy signal instances) and hence fall under the broad spectrum of unsupervised dictionary learning methods. Unsupervised dictionary learning methods rely on the structure of (unpaired) corrupted signals/images (or clean signals) to learn the underlying models. These structures of the signals can get buried when learning from signals with significant levels of noise [7]. However, supervised learning methods utilize both corrupted signals and known ground-truth signals to learn the operators. More importantly, the transforms are learned to directly minimize image quality metrics of interest, and hence can capture relevant features even in the presence of significant amounts of noise.

Learning the regularizer in a supervised fashion has been often cast as a bilevel problem in earlier works [8–12]. One of the methods to solve the bilevel problem was to replace the lower-level problem by its first order optimality conditions obtained through KKT conditions and then solve the bilevel problem using an implicit differentiation method. However, sparsity-promoting regularizers typically involve nonsmooth penalties such as with the ℓ_1 norm. Some works [13–16] used relaxed versions of the ℓ_1 norm to perform implicit differentiation. However, these smooth approximations introduce errors in calculating the gradients close to the non-differentiable points.

We propose a direct approach to solving the bilevel formulation by replacing the lower-level variational formulation by an implicit closed-form expression. While we demonstrate supervised learning of transforms for denoising ($\boldsymbol{A} = \boldsymbol{I}$), our approach can be extended to general inverse problems. Automatic differentiation packages such as CVXPY's Diffcp [17–19] allow for the exact evaluation of the Jacobian of an arbitrarily complicated differentiable function by partitioning the function into a sequence of simple operations which are by themselves trivially differentiable. We compare our approach of deriving gradients by implicit differentiation of a closed-form expression with the auto-differentiation approaches in terms of accuracy and time of computation. To our knowledge, no work to date has solved the bilevel sparse signal reconstruction problem by exploiting implicit closed-form expressions or without making relaxations. Extensive experimental results on 1D signals and images demonstrate that our method learns meaningful transforms taking into account both the dataset and the task at hand and with less runtime than the autodifferentiation methods.

2. BILEVEL FORMULATION OF SUPERVISED LEARNING AND ITS ANALYSIS

The general bilevel sparsifying transform learning formulation for denoising can be posed as in (2). The bilevel learning formulation (2)

Emails: A. Ghosh (*ghoshavr@msu.edu*), M. T. McCann (*mccann@lanl.gov*), and S. Ravishankar (*ravisha3@msu.edu*).

consists of two problems: the upper-level and the lower-level problem. The upper-level problem minimizes the cost $\mathcal{Q}(\boldsymbol{W})$ with respect to the parameters $\boldsymbol{W}$, where $\mathcal{Q}(\boldsymbol{W})$ compares the ground truth $\boldsymbol{x}_t$ with the parameterized reconstruction output $\boldsymbol{x}_t^*(\boldsymbol{W})$ obtained from the lower-level optimization problem. The lower-level problem is a variational reconstruction problem to obtain the parameterized reconstruction solution $\boldsymbol{x}_t^*(\boldsymbol{W})$ from $\boldsymbol{y}_t$.

$$\begin{aligned} \arg\min_{\boldsymbol{W}} \mathcal{Q}(\boldsymbol{W}) &= \sum_{t=1}^{T} \frac{1}{2}\|\boldsymbol{x}_t^*(\boldsymbol{W}) - \boldsymbol{x}_t\|_2^2 \\ \text{s.t. } \boldsymbol{x}_t^*(\boldsymbol{W}) &= \arg\min_{x} \|\boldsymbol{x} - \boldsymbol{y}_t\|_2^2 + ||\boldsymbol{W}\boldsymbol{x}||_1 \end{aligned} \tag{2}$$

In this supervised setting, the training samples $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ are given and denote a clean image and its noisy version. Our objective is to learn the transform matrix $\boldsymbol{W}$ so that the lower level reconstruction using this sparsity operator is as close as possible to the ground truth. It is important to note two things in the above bilevel problem: first, no constraint is imposed on $\boldsymbol{W}$ during learning; second, the algorithm learns the scaling of the regularization penalty, hence a separate scalar regularization strength is not required (i.e., $\beta||\boldsymbol{W}\boldsymbol{x}||_1 = ||\beta\boldsymbol{W}\boldsymbol{x}||_1$ for $\beta \geq 0$). As a first step towards solving (2), we derive an implicit closed-form expression for the lower-level problem.

2.1. Closed-form solution obtained by duality

Consider the lower-level functional

$$\mathcal{J}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \|\boldsymbol{W}\boldsymbol{x}\|_1, \tag{3}$$

with $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\boldsymbol{W} \in \mathbb{R}^{k \times n}$. It is strictly convex in $\boldsymbol{x}$ (the first term is strictly convex, while the second is convex), and therefore has a unique global minimizer. Thus, we can write $\boldsymbol{x}^*(\boldsymbol{W}) = \arg\min_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{x}, \boldsymbol{W})$ without the possibility of the minimizer not existing or being a nonsingleton. Note that although $\boldsymbol{x}^*$ depends on both $\boldsymbol{y}$ and $\boldsymbol{W}$, the $\boldsymbol{y}$-dependence is not relevant for this derivation and we will not include it explicitly.

A key component in deriving the closed-form expression for the minimizer of (3) is that we need to know the sign pattern of $\boldsymbol{W}\boldsymbol{x}^*(\boldsymbol{W})$. So, the closed-form expression is an implicit equation (4), where the reconstruction $\boldsymbol{x}^*(\boldsymbol{W})$ is dependent on $\text{sign}(\boldsymbol{W}\boldsymbol{x}^*)$ (where $\text{sign}(\mathbf{z})_i$ is defined to be -1 when $\mathbf{z}_i < 0$, 0 when $\mathbf{z}_i = 0$, and 1 when $\mathbf{z}_i > 0$), which itself is a function of $\boldsymbol{x}^*$. So, to obtain the closed-form expression, we first solve for $\boldsymbol{x}^*$ using any well-known iterative minimization algorithm like ADMM or PGD [20] and use it to compute the sign pattern. It is natural to question the approach of using an implicit closed-form equation if there exist well-known iterative minimization algorithms like ADMM and PGD to solve for $\boldsymbol{x}^*$. We note that our objective in this step is not to find $\boldsymbol{x}^*$ but rather to obtain a closed-form expression that allows us to take gradients with respect to $\boldsymbol{W}$.

Let $k_{=0}$ denote the set $\{i \in (1, 2, 3, .., k) \,:\, (\boldsymbol{W}\boldsymbol{x}^*(\boldsymbol{W}))_i = 0\}$ and $k_{\neq 0}$ denote $\{i \in (1, 2, 3, .., k) \,:\, (\boldsymbol{W}\boldsymbol{x}^*(\boldsymbol{W}))_i \neq 0\}$, then we have the following theorem.

Theorem 1 (Closed-form expression of $\arg\min_{\boldsymbol{x}} \mathcal{J}(\boldsymbol{x}, \boldsymbol{W})$)**.** *Let the nonzero pattern $\boldsymbol{s}$ denote $\boldsymbol{R}\,\text{sign}(\boldsymbol{W}\boldsymbol{x}^*)$ and let $\boldsymbol{W}_0$ and $\boldsymbol{W}_\pm$ contain the rows of $\boldsymbol{W}$, whose indices are given by the set $k_{=0}$ and $k_{\neq 0}$, respectively. Then, the closed-form expression for $\boldsymbol{x}^*(\boldsymbol{W}) = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \|\boldsymbol{W}\boldsymbol{x}\|_1$ is obtained from Lagrangian dual analysis [21] as*

$$\boldsymbol{x}^*(W) = \boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)}(\boldsymbol{y} - \boldsymbol{W}_\pm^\mathsf{T}\boldsymbol{s}), \tag{4}$$

where $\boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)}$ is the projector matrix onto the nullspace of $\boldsymbol{W}_0$ and is given by $\boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)} = (\boldsymbol{I} - \boldsymbol{W}_0^\dagger\boldsymbol{W}_0)$, and $\boldsymbol{R}$ is a row selection matrix that helps retain only the nonzero elements of the sign vector $\text{sign}(\boldsymbol{W}\boldsymbol{x}^*)$.

2.2. Gradient calculations

Once we have a closed-form expression of the lower level problem, our next step is to compute the derivative of $\boldsymbol{x}^*(\boldsymbol{W})$ with respect to $\boldsymbol{W}$ using matrix calculus [22]. Note that in order for the gradients to exist, the sign pattern vector $\text{sign}(\boldsymbol{W}\boldsymbol{x}^*)$ has to remain constant in an open set containing $\boldsymbol{W}$. Only then is the closed-form expression for $\boldsymbol{x}^*(\boldsymbol{W})$ valid in each region where $\text{sign}(\boldsymbol{W}\boldsymbol{x}^*)$ is constant. As a brief example of this, consider the scalar denoising problem $x^*(w) = \arg\min_x \frac{1}{2}(x-y)^2 + |wx|$ and $s(w) = \text{sign}(wx^*(w))$. Assuming that $y \geq 0$, one can show that $x^*(w) = y - |w|$ when $y - |w| \geq 0$ and 0 otherwise. As a result, $s((0, y)) = 1$, $s((-y, 0)) = -1$, and $s((-\infty, -y] \cup 0 \cup [y, \infty)) = 0$ is piecewise constant; a similar result holds when $y \leq 0$. Thus $\mathcal{Q}(w) = (x^*(w) - x_t)^2$ is smooth except at $w = 0, -y, y$. So, there exist intervals in the domain of w, where $\mathcal{Q}(w)$ is differentiable with respect to w. This can be shown for some higher dimensional cases as well.

Theorem 2 (Differential of the closed-form)**.** *Let the sign vector be denoted as $c(\boldsymbol{W}) = \text{sign}(\boldsymbol{W}\boldsymbol{x}^*(\boldsymbol{W}))$. Then if $c(\boldsymbol{W})$ is a constant vector in an open neighbourhood containing $\boldsymbol{W}$, the gradients of the closed-form expression in* (4) *wrt $\boldsymbol{W}$ exist and the form of the differential [22] is given by:*

$$\partial\boldsymbol{x}^* = -\boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)}\partial\boldsymbol{W}_\pm^\mathsf{T}\boldsymbol{s} \tag{5}$$

$$\partial\boldsymbol{x}^* = -(\boldsymbol{C} + \boldsymbol{C}^\mathsf{T})(\boldsymbol{y} - \beta\boldsymbol{W}_\pm^\mathsf{T}\boldsymbol{s}) \tag{6}$$

where $\boldsymbol{C} = \boldsymbol{W}_0^\dagger\partial\boldsymbol{W}_0\boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)}$, and $\partial\boldsymbol{x}^*$, $\partial\boldsymbol{W}_0$, and $\partial\boldsymbol{W}_\pm$ are the differentials of $\boldsymbol{x}^*$, $\boldsymbol{W}_0$, and $\boldsymbol{W}_\pm$, respectively.

Based on Theorem 2, we present the final expression of the gradient of the upper level cost $\boldsymbol{U}(\boldsymbol{W})$ with respect to $\boldsymbol{W}$.

Lemma 1. *Let $\boldsymbol{U}(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{x}^*(\boldsymbol{W}, \boldsymbol{y}_t) - \boldsymbol{x}_t\|_2^2$ be the upper-level cost function. Then by the chain rule in matrix calculus, we derive an expression for the gradient of the cost $\boldsymbol{U}(\boldsymbol{W})$ with respect to $\boldsymbol{W}_0$ and $\boldsymbol{W}_\pm$ (denoted by $\nabla_{\boldsymbol{W}_0}\boldsymbol{U}$ and $\nabla_{\boldsymbol{W}_\pm}\boldsymbol{U}$) as*

$$\nabla_{\boldsymbol{W}_\pm}\boldsymbol{U} = -\boldsymbol{s}(\nabla_{\boldsymbol{x}^*}\boldsymbol{U})^\mathsf{T}\boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)} \tag{7}$$

$$\nabla_{\boldsymbol{W}_0}\boldsymbol{U} = -(\boldsymbol{P}_{\mathcal{N}(\boldsymbol{W}_0)}(\boldsymbol{q}(\nabla_{\boldsymbol{x}^*}\boldsymbol{U})^\mathsf{T} + \nabla_{\boldsymbol{x}^*}\boldsymbol{U}\boldsymbol{q}^\mathsf{T})\boldsymbol{W}_0^\dagger)^\mathsf{T}, \tag{8}$$

with $\boldsymbol{q} = \boldsymbol{y}_t - \boldsymbol{W}_\pm^\mathsf{T}\boldsymbol{s}$. Here, $\nabla_{\boldsymbol{x}^*}\boldsymbol{U} = (\boldsymbol{x}^* - \boldsymbol{x}_t)$.

3. BLORC ALGORITHM

We propose an iterative Bilevel Learning Of ℓ_1 Regularizers with Closed-form gradients (BLORC) algorithm for (2). We perform minibatch gradient descent to learn the transform matrix $\boldsymbol{W}$ from supervised training pairs $(\boldsymbol{x}_t, \boldsymbol{y}_t)$. In all our experiments, we start from the identity matrix $\boldsymbol{W}_0 = \boldsymbol{I}_{n \times n}$. At the start of each epoch, the training pairs are randomly shuffled to remove any dataset bias. Each sample in a batch is processed as follows. **(a)** given the measurement $\boldsymbol{y}_t$ and current $\boldsymbol{W}$, the lower-level reconstruction problem is solved iteratively using ADMM to obtain an estimate of $\boldsymbol{x}_t^*(\boldsymbol{W})$. **(b)** the sign vector $\text{sign}(\boldsymbol{W}\boldsymbol{x}_t^*(\boldsymbol{W}))$ is obtained after hard-thresholding

$\boldsymbol{W}\boldsymbol{x}_t^*(\boldsymbol{W})$ with a small parameter γ. The sign vector is of paramount importance as it decides the row-split of $\boldsymbol{W}$ into $\boldsymbol{W}_0$ and $\boldsymbol{W}_\pm$. Hence, we run ADMM for a few thousands of iterations to ensure convergence of the sign pattern. **(c)** Then, using (7) and (8), we obtain the gradient of the upper level cost on a single training pair, i.e., $\nabla_{\boldsymbol{W}}\boldsymbol{U}$ which is the row-concatenation of $\nabla_{\boldsymbol{W}_0}\boldsymbol{U}$ and $\nabla_{\boldsymbol{W}_\pm}\boldsymbol{U}$. Averaging the gradients over the samples in the batch yields the minibatch gradient.

At the end of each batch, we update the matrix $\boldsymbol{W}$ based on the learning rate α and the mini-batch gradient. The updated $\boldsymbol{W}$ is used in the next batch in step **(a)** 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. Then the 2D patches were converted to 1D arrays as a pre-processing (first) step. In the end, the rows of the learned $\boldsymbol{W}$ were reshaped to look like convolutional filter patches. Except this first and last step, all the intermediate steps were the same for both 1D signals and image patches.

4. NUMERICAL EXPERIMENTS

To demonstrate how well BLORC learns the transform matrix $\boldsymbol{W}$, we perform denoising experiments on synthetic 1D signals. These synthetic 1D signals were chosen to be sparse with respect to a specific transform that also provides a baseline to compare our learned transforms with.

4.1. Denoising experiments

In our experiment, we generate $M = 4000$ training pairs $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, where the $\boldsymbol{x}_t$'s are piece-wise constant signals of length $n = 64$ (with peak value normalized to 1) and the $\boldsymbol{y}_t$'s are noisy versions with additive i.i.d. Gaussian noise with standard deviation $\sigma = 0.1$. Figure 1(a) shows a single pair of such $(\boldsymbol{x}_t, \boldsymbol{y}_t)$. 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 1(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 1(c). The learned transform is row-rearranged such that it has maximum correlation with the 1D-DCT matrix and is shown in Figure 1(d). What is interesting in the learned transforms of Figures 1(b) and 1(d) is that they exactly capture the same intuition as the standard finite difference transform and the 1D-DCT transform, respectively, but in addition, they also have some novel features learned for the denoising task. Experimental results suggest that the learned transforms perform better than both the standard transforms and analysis dictionaries learned in an unsupervised manner and applied for denoising a test set. The unsupervised analysis dictionary $(\boldsymbol{W})$ is learned by minimizing the objective $\sum_{t=1}^{T} \|\boldsymbol{W}\boldsymbol{x}_t\|_1$, ,where $\boldsymbol{x}_t$ is a clean signal. We enforce an orthogonal constraint on $\boldsymbol{W}$ to avoid trivial zero solutions. The training objective was minimized using the ADMM algorithm with split variables $\boldsymbol{z}_t = \boldsymbol{W}\boldsymbol{x}_t$ and where orthogonality on $\boldsymbol{W}$ was enforced by solving an orthogonal Procrustes problem using singular value decomposition (SVD) [23] each iteration. The average PSNR for 20 piecewise-constant test signals denoised using the BLORC learned $\boldsymbol{W}$ was 26.2 dB, with the PSNR of the noisy signals being 19.5 dB. The PSNR achieved using the standard transform and the operator learned using the unsupervised approach were worse at 25.4 dB and 25.9 dB, respectively. In both these cases, the lower-level denoising problem was solved using ADMM with a properly chosen regularization parameter.

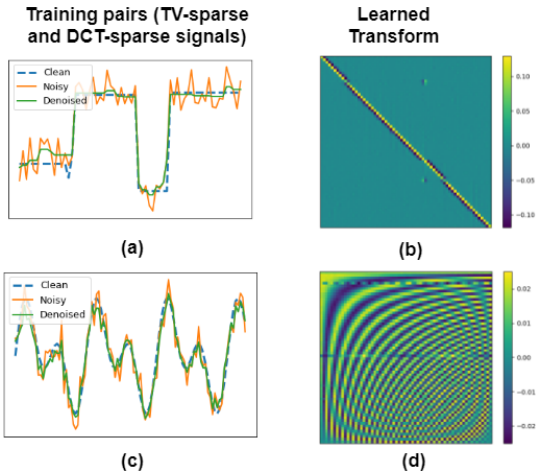

Fig. 1: 1D training pairs (x_t, y_t) (left column) and the corresponding learned transform $\hat{\boldsymbol{W}}$ (right column).

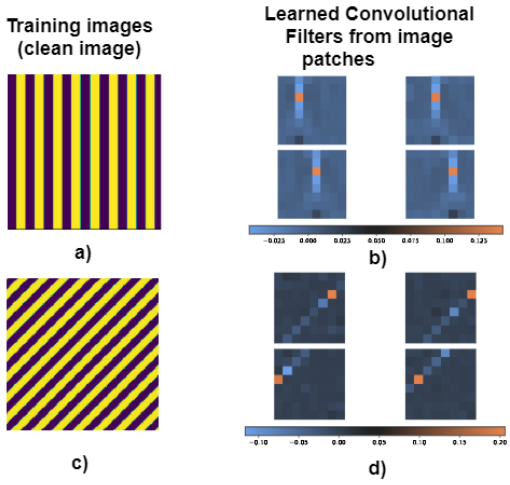

Fig. 2: Training images (only clean images being shown) with a specific orientation (left) and learned convolutional filters (right).

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 1: Time and error comparisons for Gradient calculation averaged over 100 different single training pairs (x_t, y_t).

Extending the BLORC algorithm to images, or more specifically to image patches, we learn reasonable transforms as well. We chose images (normalized) of size 256×256 with directional patterns (vertical strips and diagonal strips) and generated their noisy versions with i.i.d. Gaussian noise with $\sigma = 0.1$. We extracted image patches

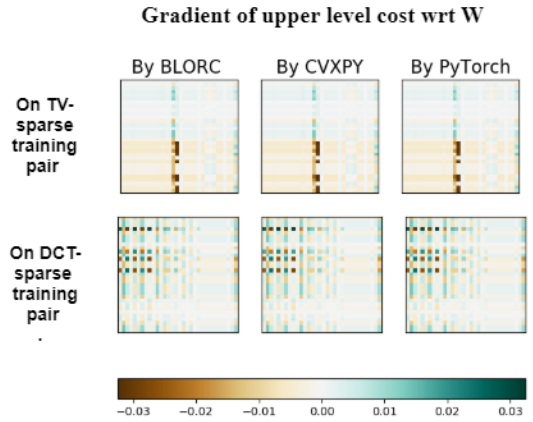

Fig. 3: Gradient of upper level cost ($\nabla_{\boldsymbol{W}}\boldsymbol{U}$) for a single training pair (x_t, y_t).

of size 8×8 with an overlap stride of 7. The reason for choosing a large stride was to ensure that distinct training pairs were obtained. As a pre-processing step, the image patches of size 8×8 were vectorized to size $n = 64$. All the parameters and the intermediate steps of the minibatch gradient descent are unchanged from the 1D signal experiment above. The learned convolutional filters in Figs. 2(b) and 2(d) look orthogonal to their image counterparts in Figs. 2(a) and 2(c), respectively, which is the expected output. In particular, these transforms are learned not to sparsify the training pairs but to minimize the gap between reconstructions and the ground truth.

4.2. Accuracy of gradient computations

While the BLORC algorithm uses an explicit form of the gradient, in automatic differentiation approaches the task is divided into a sequence of differentiable operations as computational graphs on which backpropagation is performed through 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 and plot $\nabla_{\boldsymbol{W}}\boldsymbol{U}$ for a single training pair (x_t, y_t) using three methods with the $\boldsymbol{W} = \boldsymbol{I}$ initialization in Figure 3. In the first method, we use the direct expressions in (7) and (8) to get $\nabla_{\boldsymbol{W}}\boldsymbol{U}$, which we denote as "By BLORC" in Figure 3. For the second method, we used the CVXPY package to run an iteration of optimization over (2) for the same training pair and obtained the gradient. Finally as the third method, we used PyTorch to calculate the gradient of *our proposed closed-form expression*. As the baseline for our comparisons, we calculated the numerical gradient of the upper level problem in (2) by noting the incremental change in the cost for incremental changes in each element of the matrix $\boldsymbol{W}$.

The errors for the three methods in Table 1 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 1. It takes our method $\frac{750\times4000\times12.65}{1000\times3600} \approx$ **10 hours** to complete the experiment (excluding other operations like gradient accumulation and shuffling training set) whereas for the CVXPY method, it would take **35 hours** to learn almost the same transform! The Pytorch method utilizes our derived expression for closed-form (4) to calculate gradients, hence for a fair comparison we refer the reader more to the comparison with the CVXPY method. In Figure 3, the gradient $\nabla_{\boldsymbol{W}}\boldsymbol{U}$ calculated using all three methods encapsulates the feature of the single training pair (x_t, y_t) but the one calculated using BLORC was more accurate overall as shown in Table 1.

5. CONCLUSION

In this paper, we solve the bilevel problem of learning sparsity regularizers by analytically calculating the gradients of an implicit closed-form expression. The learned regularizers are optimal for the task at hand (here, denoising) and outperform known regularizers on test data. Our mathematical analysis of bilevel reconstruction may lay the cornerstone for learning sparsifying (including deep) regularizers for inverse problems with general forward operators $\boldsymbol{A}$, which is of major interest to the computational imaging community.

6. REFERENCES

[1] D. L. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] Saiprasad Ravishankar and Yoram Bresler, "Learning doubly sparse transforms for images," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4598–4612, 2013.

[3] Saiprasad Ravishankar and Yoram Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2012.

[4] Ron Rubinstein, Tomer Peleg, and Michael Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2012.

[5] Ye Zhang, Haolong Wang, and Wenwu Wang, "An analysis dictionary learning algorithm based on recursive least squares," in *2014 12th International Conference on Signal Processing (ICSP)*. IEEE, 2014, pp. 831–835.

[6] Mehrdad Yaghoobi, Sangnam Nam, Rémi Gribonval, and Mike E Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, 2013.

[7] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[8] Gernot Holler, Karl Kunisch, and Richard C Barnard, "A bilevel approach for parameter learning in inverse problems," *Inverse Problems*, vol. 34, no. 11, pp. 115012, 2018.

[9] Karl Kunisch and Thomas Pock, "A bilevel optimization approach for parameter learning in variational models," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 938–983, Jan. 2013.

[10] Risheng Liu, Long Ma, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang, "Bilevel integrative optimization for ill-posed inverse problems," *arXiv preprint arXiv:1907.03083*, 2019.

[11] Giovanni S Alberti, Ernesto De Vito, Matti Lassas, Luca Ratti, and Matteo Santacesaria, "Learning the optimal regularizer for inverse problems," *arXiv preprint arXiv:2106.06513*, 2021.

[12] Peter Ochs, René Ranftl, Thomas Brox, and Thomas Pock, "Bilevel optimization with nonsmooth lower level problems," in *Lecture Notes in Computer Science*, pp. 654–665. Springer International Publishing, 2015.

[13] Yunjin Chen, Thomas Pock, and Horst Bischof, "Learning ℓ_1-based analysis and synthesis sparsity priors using bi-level optimization," *arXiv:1401.4105 [cs.CV]*, Jan. 2014.

[14] Pablo Sprechmann, Roee Litman, Tal Ben Yakar, Alexander M Bronstein, and Guillermo Sapiro, "Supervised sparse analysis and synthesis operators," in *Advances in Neural Information Processing Systems 26*, pp. 908–916. 2013.

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

[16] Gabriel Peyré and Jalal M. Fadili, "Learning analysis sparsity priors," in *Sampling Theory and Applications*, Singapore, Singapore, May 2011, p. 4.

[17] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. Moursi, "diffcp: differentiating through a cone program, version 1.0," `https://github.com/cvxgrp/diffcp`, 2019.

[18] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. Moursi, "Differentiating through a cone program," *Journal of Applied and Numerical Optimization*, vol. 1, no. 2, pp. 107–115, 2019.

[19] Brandon Amos, *Differentiable Optimization-Based Modeling for Machine Learning*, Ph.D. thesis, Carnegie Mellon University, May 2019.

[20] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[21] Ryan J Tibshirani and Jonathan Taylor, "The solution path of the generalized lasso," *The annals of statistics*, vol. 39, no. 3, pp. 1335–1371, 2011.

[22] Thomas P. Minka, "Old and new matrix algebra useful for statistics," Tech. Rep., MIT Media Lab, 2000.

[23] Saiprasad Ravishankar and Yoram Bresler, "ℓ_0 sparsifying transform learning with efficient optimal updates and convergence guarantees," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2389–2404, 2015.