Toward Designing Convergent Deep Operator Splitting Methods for Task-specific Nonconvex Optimization

Risheng Liu[1,2*], **Shichao Cheng**[2,3], **Yi He**[1,2], **Xin Fan**[1,2], **Zhongxuan Luo**[1,2,3]
[1]International School of Information Science & Engineering, Dalian University of Technology
[2]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province
[3]School of Mathematical Science, Dalian University of Technology
{rsliu, xin.fan, zxluo}@dlut.edu.cn, {shichao.cheng, heyiking}@outlook.com

Abstract

Operator splitting methods have been successfully used in computational sciences, statistics, learning and vision areas to reduce complex problems into a series of simpler subproblems. However, prevalent splitting schemes are mostly established only based on the mathematical properties of some general optimization models. So it is a laborious process and often requires many iterations of ideation and validation to obtain practical and task-specific optimal solutions, especially for nonconvex problems in real-world scenarios. To break through the above limits, we introduce a new algorithmic framework, called Learnable Bregman Splitting (LBS), to perform deep-architecture-based operator splitting for nonconvex optimization based on specific task model. Thanks to the data-dependent (i.e., learnable) nature, our LBS can not only speed up the convergence, but also avoid unwanted trivial solutions for real-world tasks. Though with inexact deep iterations, we can still establish the global convergence and estimate the asymptotic convergence rate of LBS only by enforcing some fairly loose assumptions. Extensive experiments on different applications (e.g., image completion and deblurring) verify our theoretical results and show the superiority of LBS against existing methods.

1 Introduction

In this work, we consider the optimization problem

$$\min_{x_n \in \mathcal{X}_n} \Psi(x) := f(x) + \sum_{n=1}^{N} g_n(x_n), \tag{1}$$

where $x = \{x_1, \cdots, x_N\} \in \mathbb{R}^D$ has N blocks ($N \geq 1$), $f(x)$ is continuously differentiable, but a series of $g_n(x)$ is not necessarily differentiable. Notice that convexity is not assumed for f, g_n or $\mathcal{X}_n$. By considering g_n as an extended value function (i.e., take $+\infty$ value), we can incorporate the set constraint $x_n \in \mathcal{X}_n$ into $g_n(x_n)$ since this is equivalent to minimize the indicator function of $\mathcal{X}_n$. Therefore, we will not include this set constraint in our following analysis. Typically, f captures the loss of data fitting from the specific task modeling and $g = \sum_{n=1}^{N} g_n$ is the regularization that promotes desired structures on the variable x. Problems appearing in many learning and vision applications, such as sparse coding [Lin *et al.*, 2011], tensor factorization [Xu and Yin, 2017], and image restoration [Liu *et al.*, 2018] can all be (re)formulated in the form of Eq. (1).

1.1 Related Works

One of the most prevalent algorithms to solve Eq. (1) is the operator splitting approach. The main idea behind such kind of schemes is to reduce complex problems built from simple pieces into a series smaller subproblems which can be solved sequentially or in parallel. In the past several decades, a variety of splitting methods have been designed and analyzed. For example, [Passty, 1979] provided a prototype of the Forward-Backward Splitting (FBS) and proved its ergodic convergence. [Beck and Teboulle, 2009] presented the convergence rate of proximal gradient (PG) and accelerated proximal gradient (APG, also known as FISTA). Recently, [Davis and Yin, 2016] provided a unified way to analyze the convergence rate of Peaceman-Rachford Splitting (PRS) and Douglas-Rachford Splitting (DRS). It is also known that the widely used Alternating Direction Method of Multipliers (ADMM) can be reformulated within the operator splitting (e.g., DRS) framework in the dual space [Lin *et al.*, 2015]. Though with mathematically proved convergence properties, the generally designed algorithms may still fail on some particular nonconvex optimization models in real scenarios. This is mainly because that due to their fixed updating schemes, it is hard to escape the unwanted saddle points during iterations.

To improve the performance in practical real-world applications, some researches tried to parameterize exiting iteration schemes and learned their parameters in the resulted propagation models. For example, [Liu *et al.*, 2016] learned the parameters of a parameterized partial differential equation for various image and video processing tasks. Similarly, [Chen *et al.*, 2015] introduced a higher-order diffusion system to perform data-dependent gradient descent for image denoising

*Corresponding Author.

and super-resolution. The studies in [Uwe and Stefan, 2014] parametrized the half-quadratic splitting for practical applications. Very recently, inspired by the success of deep networks in different application fields, some works also tried to replace the standard iterations by existing network architectures. By considering convolutional neural networks (CNNs) as special image prior, [Zhang *et al.*, 2017] proposed an iterative CNN scheme to address image restoration problems.

However, we have to point out that although with relatively better performance in some specific tasks, the nice convergence properties proved from theoretical side are completely missing in these methods. That is, neither the adaptive parameterization nor the replaced CNNs mentioned above can preserve the convergence results proved for the original iteration schemes. Moreover, it is even impossible to investigate and control the iterative behaviors (e.g., descent) of these methods, since their learned iterations actually no longer solve the original optimization model.

1.2 Contributions

In this work, we propose Learnable Bregman Splitting (LBS), a novel deep operator splitting algorithm for nonconvex optimization in real-world scenarios. Specifically, we first introduce a Bregman distance function to penalize the variables at each iteration. Then the basic LBS updating scheme is established based on a relaxed Krasonselskii-Mann iteration [Davis and Yin, 2016]. By introducing a novel triple operator splitting strategy, we can successfully combine the task-model-inspired and data-learning-driven operators within the LBS algorithmic framework. In summary, our contributions mainly include:

- LBS provides a novel learning strategy to extend prevalent mathematically designed operator splitting schemes for task-specific nonconvex optimization. Thanks to the learnable deep architectures, we can learn our iterations on collected training data to avoid unwanted solutions in particular applications.
- Different from most existing learning-based optimization algorithms (e.g., iteration parameterization and CNN incorporation methods mentioned above), in which there is no theoretical guarantee, we provide rich investigations on the iterative behaviors, prove the global convergence and estimate the convergence rate of our LBS.
- We also demonstrate how to apply our algorithm for different computer vision applications and extensive results verify that LBS outperforms state-of-the-art methods on all the compared problems.

2 Learnable Bregman Splitting Method

In this section, a learning-based operator splitting method, named Learnable Bregman Splitting (LBS), is developed for the nonconvex optimization model in Eq. (1).

2.1 Bregman Distance Penalization

As a fundamental proximity measure, the Bregman distance[1] plays important roles in various iteration algorithms. However, since it does not satisfy the triangle inequality nor symmetry, this function is not a real metric. Given a convex differential function h, the associated Bregman distance can be written as

$$\Delta_h(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle. \tag{2}$$

Clearly, Δ_h is strictly convex with respect to the first argument. Moreover, $\Delta_h \geq 0$ for all (x, y) and is equal to zero if and only if $x = y$. So Δ_h actually provides a natural (asymmetric) proximity measure between points in the domain of h.

In this work, we introduce Δ_{h_n} as a penalty term for each x_n at t-th iteration. That is, we actually minimize the following energy to update x^{t+1}:

$$\Psi_h^{t+1}(x) = f(x) + \sum_{n=1}^{N} g_n(x_n) + \frac{1}{\lambda}\Delta_h(x, x^t), \tag{3}$$

where we denote $\Delta_h(x, x^t) = \sum_{n=1}^{N} \Delta_{h_n}(x_n, x_n^t)$ and $\lambda > 0$ is the penalty parameter. It will be demonstrated that $\{\Delta_{h_n}(x_n, x_n^t)\}$ brings nice convergence properties for the proposed optimization model when h is μ-strong convex [Bauschke *et al.*, 1997].

2.2 Uniform Coordinate Updating Scheme

In this work, we consider the following general coordinate update scheme to minimize the energy function in Eq. (3):

$$x_n^{t+1} = x_n^t - \gamma^t [\mathcal{D}(x^t)]_n, \tag{4}$$

where $\mathcal{D}(x^t)$ denotes the update direction (regarding to the problem) on x^t, $\gamma^t > 0$ is a step size and $[\cdot]_n$ denotes the n-th block of the given variable. It should be pointed out that by formulating $\mathcal{D} := \mathcal{I} - \mathcal{T}$ (here $\mathcal{I}$ denotes the identity mapping), Eq. (4) can be further recognized as a relaxed Krasonselskii-Mann iteration [Shi *et al.*, 2016] with the operator $\mathcal{T}$ (i.e., $x_n^{t+1} = (1 - \gamma^t)x_n^t + \gamma^t[\mathcal{T}(x^t)]_n$) and then various existing first-order schemes can be reformulated in the form of Eq. (4).

Specifically, by defining $\mathcal{J}_{\mathcal{F}} = (\mathcal{I} - \mathcal{F})^{-1}$ (resolvent), $\mathcal{R}_{\mathcal{F}} = 2\mathcal{J}_{\mathcal{F}} - \mathcal{I}$ (reflection) for $\mathcal{F}$ (operator about f), we can obtain a variety of prevalent splitting schemes, such as FBS, PRS, and DRS. As for the operator $\mathcal{T}$ in our work, if setting $\mathcal{T} = \mathcal{I} - \mathcal{J}_{\mathcal{G}} \circ (\mathcal{I} - \mathcal{F})$ and $\gamma^t = 1$, we obtain FBS from Eq. (4), i.e., $x^{t+1} = \mathcal{J}_{\mathcal{G}} \circ (\mathcal{I} - \mathcal{F})(x^t)$, where $\circ$ denotes the operator composition. By considering $\mathcal{J}_{\mathcal{G}} = \texttt{prox}_g$[2] and $\mathcal{F} = \nabla f$, we further have the well-known proximal (or projected) gradient scheme from FBS. Setting $\mathcal{T} = \mathcal{R}_{\mathcal{G}} \circ \mathcal{R}_{\mathcal{F}}$ and $\gamma^t = 1$, Eq. (4) reduces to $x^{t+1} = (\mathcal{R}_{\mathcal{G}} \circ \mathcal{R}_{\mathcal{F}})(x^t)$, which is just the standard PRS iteration. Similarly, with the same $\mathcal{T}$ in PRS and $\gamma = 1/2$, we can also deduce DRS.

Additionally, it should be pointed out that the well-known ADMM [Lin *et al.*, 2011] can also be deduced by applying DRS on its Lagrange dual space [Davis and Yin, 2016]. Therefore, although the original ADMM is designed for linearly constraint models, we can still reformulate it as a special case of Eq. (4) in the dual variable space. Thus Eq. (4) actually can also be utilized to address the constrained problems.

[1] The use of Bregman distance in optimization within various contexts is well spread. Many interesting properties of this function can be found in the comprehensive work [Bauschke *et al.*, 1997].

[2] The proximal operation with respect to g (denoted as $\texttt{prox}_g$) is defined as $\texttt{prox}_g(x) := \arg\min_y g(y) + \frac{1}{2}\|y - x\|^2$.

In consideration of the nice properties and high flexibility of Eq. (4) mentioned above, we would like to utilize the general updating scheme in Eq. (4) with $\mathcal{D} = \mathcal{I} - \mathcal{T}$, (named Uniform Coordinate Updating Scheme, UCUS for short) as our fundamental block-wise iteration rule.

2.3 Splitting with Learnable Architecture

As discussed above, most existing splitting algorithms (e.g., FBS, PRS and DRS) specify the operator only based on the optimization model. However, due to the nonconvex nature of the model, it is hard for these schemes to escape undesired local minimum. Moreover, the complex data distribution in real applications will also slow down the redesigned iterations.

To partially address these issues, we provide a new splitting strategy, in which a learnable operator $\mathcal{T}_d$ is introduced to extract information from the data. That is, we consider the following triple splitting scheme:

$$[\mathcal{T}(x)]_n := \mathcal{T}_{g_n} \circ \mathcal{T}_{f_\lambda} \circ \mathcal{T}_d(x), \ (n = 1, 2, \cdots, N), \quad (5)$$

where $\mathcal{T}_{g_n}$ and $\mathcal{T}_{f_\lambda}$ are operators related to Ψ_h^t in Eq. (3). Here we just follow a FBS-like strategy to define $\mathcal{T}_{f_\lambda} = \mathcal{I} - \nabla(f + \frac{1}{\lambda}\Delta_h)$ and $\mathcal{T}_{g_n} = \texttt{prox}_{\rho g_n}$. As for $\mathcal{T}_d$, we would like to build it as a learnable network architecture and train its parameters from collected training data set[3]. In this way, we can successfully incorporate data information to improve the iterative performance of the proposed algorithm.

Notice that it is challenging to analyze the convergence issues for the existing network-incorporated iterations (e.g., [Zhang *et al.*, 2017]), since all their schemes are built in heuristic manners. *In contrast, we will demonstrate in the following section that the convergence of our LBS can be strictly proved.*

2.4 The Complete Algorithm

It can be seen that the learnable operator $\mathcal{T}_d$ are not deduced from strict optimization rule, there may exists iteration errors when calculating $\mathcal{T}$ at each stage. Thus we introduce a new condition to control the inexactness of our updating scheme at each iteration. Specifically, we define the optimality errors of a given variable u at t-th iteration based on the first order subdifferential of Ψ_h^{t+1}, i.e.,

$$\begin{aligned} e_{u_n}^{t+1} := & \, d_{g_n} + \nabla_n f(\{x_{n^-}^{t+1}, u_n, x_{n^+}^t\}) \\ & + \tfrac{1}{\lambda^t}(\nabla h_n(u_n) - \nabla h_n(x_n^t)), \end{aligned}$$

where $d_{g_n} \in \partial g_n(u_n)$ (here we denote ∂g_n as the limiting Ferchet subdifferential of g_n [Xu and Yin, 2017]) and $n = 1, 2, \cdots, N$. Then we consider the following so-called Relaxed Optimality Condition (ROC) for the given $u \in \mathbb{R}^D$.

Condition 1. *(Relaxed Optimality Condition) Given any $u \in \mathbb{R}^D$, we define the relaxed optimality condition of Ψ_h^{t+1} for $u_n \in u$ $(n = 1, 2, \cdots, N)$ as $\|e_{u_n}^{t+1}\| \le c\|u_n - x_n^t\|$, where c is a fixed positive constant.*

Based on the above condition, we are ready to propose our LBS algorithm for solving Eq. (1) in Alg. 1. Notice that the UCUS iteration, denoted as $\texttt{ucus}(\cdot)$, is independently stated in Alg. 2. It can be seen that if ROC is satisfied, the LBS iterations are fully based on the learnable network operator. While for some iterations, which do not satisfy ROC, we may still perform the model-based operators $\mathcal{T}_{g_n} \circ \mathcal{T}_{f_{\lambda^t}}$ to guarantee the final convergence. For convenience, hereafter the subvectors $\{x_1, \ldots, x_{n-1}\}$ and $\{x_{n+1}, \ldots, x_N\}$ are denoted as x_{n^-} and x_{n^+} for short, respectively. We also denote $\psi_n^{t+1}(x_n) = f_n^{t+1}(x_n) + g_n(x_n)$, in which $f_n^{t+1}(x_n) = f(\{x_{n^-}^{t+1}, x_n, x_{n^+}^t\})$.

Algorithm 1 Learnable Bregman Splitting (LBS)

Require: x^0, $\mathcal{T}_d$, Δ_h, c, μ, $\{\gamma^t | 0 < \gamma^t \le 1\}$, $\{\lambda^t | \lambda^t > 0\}$.
while not converged **do**
  $z^{t+1} = \mathcal{T}_{f_{\lambda^t}} \circ \mathcal{T}_d(x^t)$.
  for $n = 1, 2, \ldots, N$ **do**
    $u_n^{t+1} = \mathcal{T}_{g_n}(z_n^{t+1})$.
    if u_n^{t+1} satisfies ROC **then**
      $v_n^{t+1} = u_n^{t+1}$.
    else
      $v_n^{t+1} = \mathcal{T}_{g_n} \circ \mathcal{T}_{f_{\lambda^t}}(\{x_{n^-}^{t+1}, x_n^t, x_{n^+}^t\})$.
    end if
    $x_n^{t+1} = \texttt{ucus}(\{x_{n^-}^{t+1}, v_n^{t+1}, x_{n^+}^t\}, \gamma^t)$.
  end for
end while

Algorithm 2 $x_n^{t+1} = \texttt{ucus}(\{x_{n^-}^{t+1}, v_n^{t+1}, x_{n^+}^t\}, \gamma^t)$

$w_n^{t+1} = x_n^t - \gamma^t(x_n^t - v_n^{t+1})$.
if $\psi_n^{t+1}(w_n^{t+1}) \le \psi_n^{t+1}(v_n^{t+1})$ **then**
  $x_n^{t+1} = w_n^{t+1}$.
else
  $x_n^{t+1} = v_n^{t+1}$.
end if

3 Convergence Analysis

In this section, we provide strict analysis on the convergence behaviors of LBS. The following assumptions on the functions f, g, and Ψ are necessary for our analysis. Notice that all these assumptions are fairly loose in optimization area and satisfied in most vision and learning problems.

Assumption 1. *1) f is Lipschitz smooth and g_n is proximable[4]. 2) Ψ is coercive.*

We first summarize the roadmap of our analysis as follows[5]. Firstly, some important iterative properties of the sequences are proved in Propositions 1, 2, and 3. Based on these results, we prove the global convergence of LBS in Theorem 1. That is, the sequences generated by LBS are Cauchy sequences and converge to the critical points of Eq. (1). Finally, the convergence rate of the sequences is estimated in Corollary 1.

Proposition 1. *(Sufficient descent). If $c < \frac{\mu}{2\lambda^t}$ and $\rho < \frac{1}{L}$, both the learnable operators $\mathcal{T}_{g_n} \circ \mathcal{T}_{f_{\lambda^t}} \circ \mathcal{T}_d$ and $\mathcal{T}_{g_n} \circ \mathcal{T}_{f_{\lambda^t}}$*

[3] See Sec. 4 for the details of this operator and its training strategy.

[4] A function g is proximable if it is easy to obtain the minimizer of $g(x) + \frac{1}{2\beta}\|x - y\|$ for any given y and $\beta > 0$.

[5] The detailed proofs of the theoretical results can be found in our technical report: https://arxiv.org/abs/1804.10798.

in Alg. 1 can get the objective inequality:

$$\psi_n^{t+1}\left(v_n^{t+1}\right) \leq \psi_n^{t+1}\left(x_n^t\right) - M\|u^t - x^t\|^2,$$

where $M = \min_t\{\frac{\mu}{2\lambda^t} - c, \frac{1}{2\rho} - \frac{L}{2}\}$, *$L$ is Lipschitz moduli of ∇f. Together with the direct comparison of function values in Alg. 2, there exists a non-increasing objective sequence, i.e.,*

$$\psi_n^{t+1}(x_n^{t+1}) \leq \psi_n^{t+1}(v_n^{t+1}) \leq \psi_n^{t+1}(x_n^t).$$

Remark 1. *The inequalities in Proposition 1 builds the relationship of $\psi_n^{t+1}(x_n^t)$ and $\psi_n^{t+1}(v_n^{t+1})$, thus we can obtain a series of useful inequalities:*

$$\begin{aligned}&\Psi(x^{t+1}) = \Psi(x_{N+}^{t+1})\\ &\leq \Psi(\{x_{n-}^{t+1}, v_n^{t+1}, x_{n+}^t\}) \leq \Psi(\{x_{n-}^{t+1}, x_n^{t+1}, x_{n+}^t\})\\ &\leq \Psi(\{v_1^{t+1}, x_{1+}^t\}) \leq \Psi(\{x_1^t, x_{1+}^t\}) = \Psi(x^t),\end{aligned}$$

where $n = N-1, \ldots, 1$. It implies the non-increasing property of $\{\Psi(x^t)\}_{t\in\mathbb{N}}$.

Proposition 2. *(Square summable). If $\gamma^t \in (0, 1]$, $\{x^t\}_{t\in\mathbb{N}}$, $\{v_n^t\}_{t\in\mathbb{N}}$ are the sequences by Alg. 1, we have*

$$\sum_{t=0}^{\infty}\|x^{t+1} - x^t\|^2 \leq \sum_{t=0}^{\infty}\sum_{n=1}^{N}\|v_n^{t+1} - x_n^t\|^2 < \infty.$$

Proposition 3. *(Subsequence convergence). Let $\{x^t\}_{t\in\mathbb{N}}$ be the sequence generalized by Alg. 1. If x^* is any accumulation point of $\{x^t\}_{t\in\mathbb{N}}$. Then we have*

$$x^{t_j} \to x^*, v^{t_j} \to x^*, \lim_{j\to\infty}\Psi(v^{t_j}) = \Psi(x^*),$$

when $j \to \infty$.

Remark 2. *Indeed, Propositions 2 and 3 are the key points to prove that the sequence $\{x^t\}_{t\in\mathbb{N}}$ has critical points. Proposition 3 can be derived by combining the Lipschitz smoothness of f, lower semi-continuous of g_n, μ-strong convexity of h, and property of the learnable operators $\mathcal{T}_{g_n} \circ \mathcal{T}_{f_{\lambda^t}} \circ \mathcal{T}_d$ and $\mathcal{T}_{g_n} \circ \mathcal{T}_{f_{\lambda^t}}$ in Steps 4 and 8 of Alg. 1.*

Theorem 1. *(Critical point and Cauchy sequence). Let Assumption 1 hold for Eq.* (1)*, then the sequences $\{x^t\}_{t\in\mathbb{N}}$ generated by Alg. 1 has critical points x^* of Ψ, i.e., if Ψ^* is the limit of sequence $\{\Psi(x^t)\}_{t\in\mathbb{N}}$, We have*

$$\Psi(x^*) = \Psi^*, 0 \in \partial\Psi(x^*).$$

If Ψ is a Kurdyka–Łojasiewicz function[6]*, we can further prove that $\{x^t\}_{t\in\mathbb{N}}$ is a Cauchy sequence, thus globally converges to a critical point of Ψ.*

Based on the above theorem, we can estimate convergence rate as follows.

Corollary 1. *Let $\phi(s) = \frac{t}{\theta}(s)^\theta$ be a desingularizing function with a constant $t > 0$ and a parameter $\theta \in (0, 1]$. Then $\{x^t\}_{t\in\mathbb{N}}$ generated by Alg. 1 converges after finite iterations if $\theta = 1$. The linear and sub-linear rates can be obtained if choosing $\theta \in [1/2, 1)$ and $\theta \in (0, 1/2)$, respectively.*

[6]It should be pointed out that many functions arising in learning and vision areas, including ℓ_0 norm and rational ℓ_p norms (i.e., $p = p_1/p_2$) are all Kurdyka–Łojasiewicz functions [Liu *et al.*, 2018].

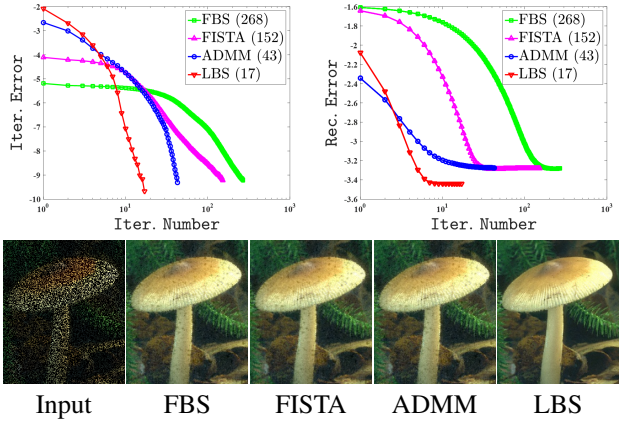


Figure 1: The iteration behaviors and visual results of LBS with comparisons to various splitting approaches, including FBS, FISTA, and ADMM. We compare the "`Iter. Error`" and "`Rec. Error`" of them on the top row and indicate the complete number of iterations in brackets after each method in the legend. The second row is the final restored results of all compared methods. The quantitative scores (PSNR / SSIM) of these methods are 25.27 / 0.65 (FBS), 25.36 / 0.65 (FISTA), 25.25 / 0.64 (ADMM), 28.45 / 0.83 (LBS), respectively.

4 Numerical Results

To verify the convergence and performance of LBS for nonconvex optimization, we apply it on two widely researched vision problems, i.e., image completion and deblurring. In our algorithm, we adopt residual network as the learnable network architecture for $\mathcal{T}_d$, which can well describe the sparse priors. Specially, there are 19 layers in our network which includes 7 convolution layers, 6 ReLU layers, 5 batch normalization layers and one loss layer. Every convolution layer has 64 kernels of size 3×3, and possesses the dilation attribute. In training stage, we randomly select 800 natural images from ImageNet database [Deng *et al.*, 2009]. The chosen pictures are cropped into small patches of size 35×35 and Gaussian noise is imposed to these patches. In the following, we always consider Mahalanobis distance $\|\cdot\|_{\mathbf{A}}$ (with weight matrix $\mathbf{A}$) [Bauschke *et al.*, 1997] as Δ_h in our applications. All experiments are performed on a PC with Intel Core i7 CPU @ 3.4 GHz, 32 RAM and NVIDIA GeForce GTX 1050 Ti GPU.

4.1 ℓ_p-Sparse Coding for Image Completion

We first consider to solve a ℓ_p-sparse coding model to address the problem of image completion (also known as image inpainting). The purpose of this task is to restore a visually plausible image in which data are missing due to damage or occlusions. This problem can be formulated as:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2\rho}\|\mathbf{M} \odot \mathbf{B}\boldsymbol{\alpha} - \mathbf{y}\|^2 + \|\boldsymbol{\alpha}\|_p^p, \tag{6}$$

where $\mathbf{y}$ is the observed image, $\mathbf{M}$ denotes a mask, $\mathbf{B}$ is the dictionary, $\boldsymbol{\alpha}$ is its corresponding sparse coefficients and $\rho > 0$ is a parameter. Following [Beck and Teboulle, 2009], we consider $\mathbf{B}$ as a inverse wavelet basis (i.e., multiplying by $\mathbf{B}$ corresponds to performing inverse wavelet transform) and thus $\mathbf{B}\boldsymbol{\alpha}$ is just the latent image (denoted as $\mathbf{x}$). To enforce the sparsity of $\boldsymbol{\alpha}$, we utilize nonconvex $\ell_{0.8}$ norm in the above

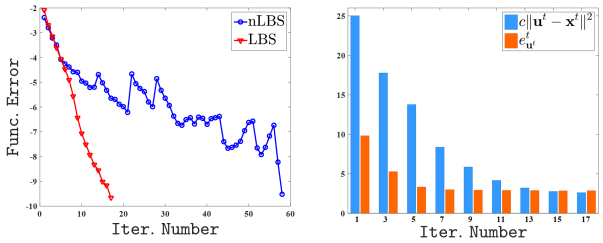


Figure 2: The left subfigure plots the "Func. Error" curves of naive LBS (nLBS) and LBS. The right subfigure illustrates when the variables in our algorithm satisfy the ROC criterion during iterations.

coding model. As for the Bergman distance, we set $\mathbf{A} = \mu\mathbf{I}$, where $\mathbf{I}$ denotes the unit matrix and $\mu = 0.01$.

It is easy to check that Eq. (6) is just a specific case of Eq. (1) with single variable. In this following, we first verify the theoretical results proposed in this work, and then test the performance of LBS on challenging benchmark datasets.

Iteration Behaviors Analysis: We first choose example images from CBSD68 dataset [Zhang *et al.*, 2017] to demonstrate the iterative behaviors of LBS together with some other widely used splitting schemes (e,g, FBS, FISTA, and ADMM). For fair comparisons, the stopping criterion of all the compared methods are set in the same manner. That is, we denote $\mathbf{x} = \mathbf{B}\boldsymbol{\alpha}$ and consider $\|\mathbf{x}^{t+1} - \mathbf{x}^t\|/\|\mathbf{x}^t\| \leq 10^{-4}$ as the stopping criterion in all these methods.

Fig. 1 showed the convergence curves from different aspects, including iteration error ("Iter. Error", defined as $\log(\|\mathbf{x}^{t+1} - \mathbf{x}^t\|/\|\mathbf{x}^t\|)$) and reconstruction error ("Rec. Error", defined as $\log(\|\mathbf{x}^t - \mathbf{x}_{gt}\|/\|\mathbf{x}_{gt}\|)$). Our LBS have superiority against traditional FBS, FISTA, and ADMM on both convergence rate and final reconstruction. LBS can achieve the convergence precision only almost a dozen steps while FBS and FISTA need few hundreds steps and ADMM needs four dozens of steps. Since introducing the network as $\mathcal{T}_d$, our strategies have less reconstruction error than others obviously. The PSNR and SSIM of the final results also verify that our LBS has better performance. Concretely, our PSNR is approximately higher 3dB than the compared methods.

We also compared the curves of objective function value errors ("Func. Error", based on $\Phi(\mathbf{x}^t)$) for different settings of LBS, including naive LBS (nLBS, do not check the ROC and monotone conditions) and the complete LBS in Alg. 1. From the left subfigure of Fig. 2, it is easy to observe that the proposed criterion can lead to very fast convergence, while there are severe oscillations on the curves of nLBS. Furthermore, we plotted the bars of ROC (i.e., the error $e^t_{\mathbf{u}^t}$ and the threshold $c\|\mathbf{u}^t - \mathbf{x}^t\|^2$) on the right part of Fig. 2. It can be seen that the ROC condition is always satisfied except at the last two iterations. Thus deep networks are performed at most of our iterations. Only at the last stages, LBS tended to perform model-inspired iterations (i.e., Step 8 in Alg. 1) to obtain accurate solution for the given optimization model.

Comparisons on Benchmarks: To further express the superiority of LBS, we generated random masks of different levels (including 20%, 40%, 60% and 80% missing pixels) on CBSD68 dataset [Zhang *et al.*, 2017] for comparison,

%	Metric	FoE	VNL	ISDSB	JSM	Ours
20	PSNR	38.23	28.87	35.20	37.55	**38.77**
	SSIM	0.95	0.95	0.96	0.98	**0.98**
40	PSNR	34.01	27.55	31.32	33.54	**34.54**
	SSIM	0.90	0.91	0.91	0.94	**0.95**
60	PSNR	30.81	26.13	28.23	29.96	**31.27**
	SSIM	0.81	0.85	0.83	0.81	**0.90**
80	PSNR	27.64	24.23	24.92	27.32	**27.71**
	SSIM	0.65	0.75	0.70	0.79	**0.80**
-	TIME	34.85	1515.49	28.00	207.57	**1.40**

Table 1: Averaged image completion performance with different levels of missing pixels on CBSD68 dataset. The percents of missing pixels are reported in the first column. TIME in the bottom row denotes the averaged run time (in seconds) on all the test images.

which contains 68 images with the size of 481×321. Then we compared LBS with four state-of-the-art methods, namely, FoE [Roth and Black, 2009], VNL [Arias *et al.*, 2011], ISDSB [He and Wang, 2014], and JSM [Zhang *et al.*, 2014]. Tab. 1 reports the averaged quantitative results, including PSNR, SSIM, and time (in second). It can be seen that regardless the proportion of masks, LBS can achieve better performance against the state-of-the-art approaches. This is mainly due to our superior strategy which uses learnable network operator.

We then compared the visual performance of LBS with all these methods. Fig. 3 presented the comparisons on an image from ImageNet database [Deng *et al.*, 2009] with 60% missing pixels. It can be seen that LBS outperformed all the compared methods on both visualization and metrics (PSNR and SSIM). The edge of motorcycle wheels can be restored smoother and clearer by LBS, while other approaches exist some noises and masks to affect the visual effects.

4.2 Nonconvex TV for Image Deblurring

We further evaluate LBS on image deblurring, which is a challenging problem in computer vision area. Here we consider the following widely used total variation (TV) based formulation:

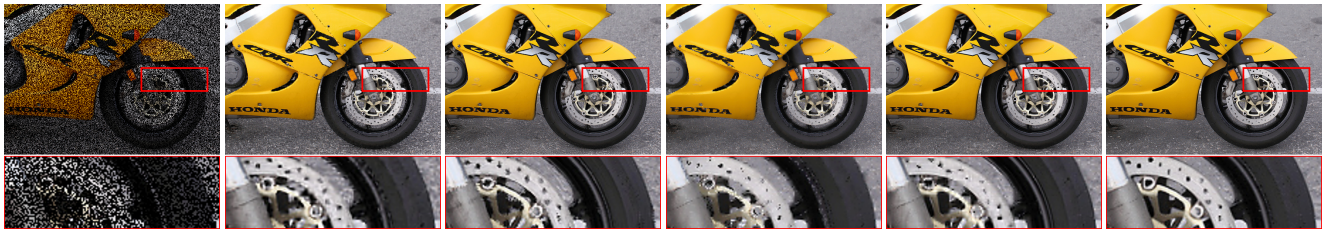
$$\min_{\mathbf{u}} \frac{1}{2\rho}\|\mathbf{k} \otimes \mathbf{u} - \mathbf{y}\|^2 + \mathtt{TV}_p(\mathbf{u}) + \chi_{\Omega_{\mathbf{u}}}(\mathbf{u}), \tag{7}$$

where $\mathbf{k}, \mathbf{x}, \mathbf{y}$ denote the blur kernel, latent image, and blurry observation, respectively. $\mathtt{TV}_p(\mathbf{u}) = \|\mathbf{D}_h\mathbf{u}\|_p^p + \|\mathbf{D}_v\mathbf{u}\|_p^p$ is the nonconvex TV regularization with gradient matrices $\mathbf{D}_h$ and $\mathbf{D}_v$ (here we also set $p = 0.8$ for the ℓ_p norm). $\chi_{\Omega_{\mathbf{u}}}(\mathbf{u})$ is the indicator function of the set $\Omega_{\mathbf{u}} := \{\mathbf{u} \in \mathcal{R}^n : 0 \leq \mathbf{u}_i \leq 1\}$. Following the half-quadratic splitting technique, Eq. (7) (with auxiliary variables $\mathbf{v}_h$ and $\mathbf{v}_v$) can be reformulated as

$$\begin{aligned}&\min_{\mathbf{u},\mathbf{v}_h,\mathbf{v}_v} \tfrac{1}{2\rho}\|\mathbf{k} \otimes \mathbf{u} - \mathbf{y}\|^2 + \|\mathbf{v}_h\|_p^p + \|\mathbf{v}_v\|_p^p \\ &+\chi_{\Omega_{\mathbf{u}}}(\mathbf{u}) + \tfrac{1}{2\eta}\left(\|\mathbf{D}_h\mathbf{u} - \mathbf{v}_h\|^2 + \|\mathbf{D}_v\mathbf{u} - \mathbf{v}_v\|^2\right).\end{aligned} \tag{8}$$

Obviously, Eq. (8) is a special case of Eq. (1) with three blocks, thus can be efficiently addressed by LBS. Here we define $\mathbf{A} = \mathrm{diag}\left(\mu_{\mathbf{u}}\mathbf{I}_{\mathbf{u}}, \mu_{\mathbf{v}_h}\mathbf{I}_{\mathbf{v}_h}, \mu_{\mathbf{v}_v}\mathbf{I}_{\mathbf{v}_v}\right)$ with $\mu_{\mathbf{u}} = 0.01$, and $\mu_{\mathbf{v}_h} = \mu_{\mathbf{v}_v} = 0.001$ in Bergman distance Δ_h.

Fig. 4 demonstrated the convergence behaviors of LBS on $\{\mathbf{u}, \mathbf{v}_h, \mathbf{v}_v\}$. It can be seen from the left subfigure that "Iter. Error" of all blocks quickly decreased to $\log(10^{-3})$,



Input - | FoE (24.45 / 0.86) | VNL (24.92 / 0.86) | ISDSB (23.27 / 0.83) | JSM (25.40 / 0.87) | LBS (**26.11 / 0.88**)

Figure 3: The inpainting results (with 60% missing pixels) of LBS with comparisons to state-of-the-art methods. The quantitative scores (i.e., PSNR / SSIM) are reported below each image.

Input - | RTF (22.78 / 0.75) | IRCNN (25.00 / 0.77) | FDN (23.79 / 0.78) | PADNet (24.39 / 0.81) | LBS (**25.34 / 0.83**)

Figure 5: The deblurring results (by a large scale blur kernel with the size 75×75) of LBS with comparisons to state-of-the-art methods. The quantitative scores (PSNR / SSIM) are reported below each method.

Metric	TV	HL	CSF	IDDBM3D	EPLL	RTF	MLP	IRCNN	FDN	PADNet	Ours
PSNR	30.67	31.03	31.55	30.79	32.44	32.45	31.47	32.61	32.65	32.69	**32.90**
SSIM	0.85	0.85	0.87	0.87	0.88	0.89	0.86	0.89	0.89	0.89	**0.90**

Table 2: Averaged quantitative results of image deblurring on the Sun *et al*. benchmark image set.

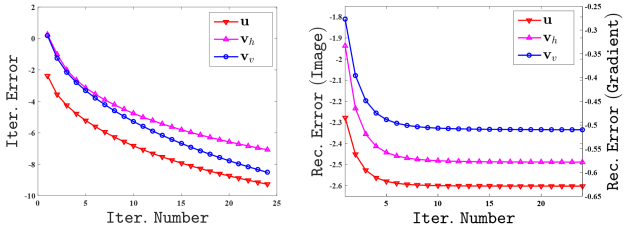


Figure 4: The iteration curves of the multiple variables in LBS. In the right subfigure, the left and right vertical ordinates are the `Rec. Error` of image (i.e., $\mathbf{u}$) and gradients (i.e., $\mathbf{v}_h$ and $\mathbf{v}_v$) respectively.

notice that "`Iter. Error`" of $\mathbf{u}$ is even less than $\log(10^{-4})$. On the right subfigure, the "`Rec. Error`" of $\mathbf{v}_h$ and $\mathbf{v}_v$ also have dramatic decline trend, which are shown along with the right vertical ordinate. Due to the different range of values, we have to plot the curves of $\mathbf{u}$ w.r.t. the left vertical ordinate. We can see that it still obtained the least "`Rec. Error`".

We then reported results on the challenging image deblurring benchmark dataset collected by Sun *et al*. [Sun *et al.*, 2013] (which includes 640 blurry images with 1% Gaussian noises) for quantitative evaluation. We compared LBS with plenty of competitive approaches, including TV [Wang *et al.*, 2008], HL [Krishnan and Fergus, 2009], CSF [Uwe and Stefan, 2014], IDDBM3D [Danielyan *et al.*, 2012], EPLL [Zoran and Weiss, 2011], RTF [Schmidt *et al.*, 2016], MLP [Schuler *et al.*, 2013], IRCNN [Zhang *et al.*, 2017], FDN [Kruse *et al.*, 2017], and PADNet [Liu *et al.*, 2018]).

It is known that learning-based methods (e.g., CSF, RTF, MLP, IRCNN, FDN, and PADNet) can achieve better performance than other conventional approaches in terms of quantitative metrics (e.g., PSNR and SSIM). However, due to the weak theoretical guarantee, they are worse than LBS (see Tab. 2). Fig. 5 expressed the qualitative results of LBS against other methods (top 4 in Tab. 2) on an example blurry image, which is generated with a large scale blur kernel (75×75 pixels) on an image from ImageNet [Deng *et al.*, 2009]. It can be seen that LBS can restore the text and windows more distinctly than others. Although IRCNN has relatively higher PSNR than others (but lower than LBS), its visual quality and SSIM are not satisfied.

5 Conclusions

This paper proposed Learnable Bregman Splitting (LBS), a novel deep architecture based operator splitting algorithm for

task-specific nonconvex optimization. It is demonstrated that both the model-based operators and the data-dependent networks can be used in our iteration. We also provided solid theoretical analysis to guarantee the convergence of LBS. The experimental results verified that LBS can obtain better performance against most other state-of-the-art approaches.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Nos. 61672125, 61733002, 61572096, 61432003 and 61632019), and the Fundamental Research Funds for the Central Universities.

References

[Arias *et al.*, 2011] Pablo Arias, Gabriele Facciolo, Vicent Caselles, and Guillermo Sapiro. A variational framework for exemplar-based image inpainting. *IJCV*, 93(3):319–347, 2011.

[Bauschke *et al.*, 1997] Heinz H Bauschke, Jonathan M Borwein, et al. Legendre functions and the method of random bregman projections. *Journal of Convex Analysis*, 4(1):27–67, 1997.

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[Chen *et al.*, 2015] Yunjin Chen, Wei Yu, and Thomas Pock. On learning optimized reaction diffusion processes for effective image restoration. In *CVPR*, 2015.

[Danielyan *et al.*, 2012] Aram Danielyan, Vladimir Katkovnik, and Karen Egiazarian. Bm3d frames and variational image deblurring. *IEEE TIP*, 21(4):1715–1728, 2012.

[Davis and Yin, 2016] Damek Davis and Wotao Yin. Convergence rate analysis of several splitting schemes. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 115–163. 2016.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[He and Wang, 2014] Liangtian He and Yilun Wang. Iterative support detection-based split bregman method for wavelet frame-based image inpainting. *IEEE TIP*, 23(12):5470–5485, 2014.

[Krishnan and Fergus, 2009] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009.

[Kruse *et al.*, 2017] Jakob Kruse, Carsten Rother, and Uwe Schmidt. Learning to push the limits of efficient fft-based image deconvolution. In *ICCV*, 2017.

[Lin *et al.*, 2011] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *NIPS*, 2011.

[Lin *et al.*, 2015] Zhouchen Lin, Risheng Liu, and Huan Li. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. *Machine Learning*, 99(2):287, 2015.

[Liu *et al.*, 2016] Risheng Liu, Guangyu Zhong, Junjie Cao, Zhouchen Lin, Shiguang Shan, and Zhongxuan Luo. Learning to diffuse: A new perspective to design pdes for visual analysis. *IEEE TPAMI*, 38(12):2457–2471, 2016.

[Liu *et al.*, 2018] Risheng Liu, Xin Fan, Shichao Cheng, Xiangyu Wang, and Zhongxuan Luo. Proximal alternating direction network: A globally converged deep unrolling framework. In *AAAI*, 2018.

[Passty, 1979] Gregory B Passty. Ergodic convergence to a zero of the sum of monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 72(2):383–390, 1979.

[Roth and Black, 2009] Stefan Roth and Michael J. Black. Fields of experts. *IJCV*, 82(2):205–229, 2009.

[Schmidt *et al.*, 2016] Uwe Schmidt, Jeremy Jancsary, Sebastian Nowozin, Stefan Roth, and Carsten Rother. Cascades of regression tree fields for image restoration. *IEEE TPAMI*, 38(4):677–689, 2016.

[Schuler *et al.*, 2013] Christian J Schuler, Harold Christopher Burger, Stefan Harmeling, and Bernhard Scholkopf. A machine learning approach for non-blind image deconvolution. In *CVPR*, 2013.

[Shi *et al.*, 2016] Hao-Jun Michael Shi, Shenyinying Tu, Yangyang Xu, and Wotao Yin. A primer on coordinate descent algorithms. *arXiv:1610.00040*, 2016.

[Sun *et al.*, 2013] Libin Sun, Sunghyun Cho, Jue Wang, and James Hays. Edge-based blur kernel estimation using patch priors. In *ICCP*, 2013.

[Uwe and Stefan, 2014] Schmidt Uwe and Roth Stefan. Shrinkage fields for effective image restoration. In *CVPR*, 2014.

[Wang *et al.*, 2008] Yilun Wang, Junfeng Yang, Wotao Yin, and Yin Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[Xu and Yin, 2017] Yangyang Xu and Wotao Yin. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, pages 1–35, 2017.

[Zhang *et al.*, 2014] Jian Zhang, Debin Zhao, Ruiqin Xiong, Siwei Ma, and Wen Gao. Image restoration using joint statistical modeling in a space-transform domain. *IEEE TCSVT*, 24(6):915–928, 2014.

[Zhang *et al.*, 2017] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *CVPR*, 2017.

[Zoran and Weiss, 2011] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.