

# Hierarchical Optimization-Derived Learning

Risheng Liu, *Member, IEEE*, Xuan Liu, Shangzhi Zeng, Jin Zhang, and Yixuan Zhang

**Abstract**—In recent years, by utilizing optimization techniques to formulate the propagation of deep model, a variety of so-called Optimization-Derived Learning (ODL) approaches have been proposed to address diverse learning and vision tasks. Although having achieved relatively satisfying practical performance, there still exist fundamental issues in existing ODL methods. In particular, current ODL methods tend to consider model constructing and learning as two separate phases, and thus fail to formulate their underlying coupling and depending relationship. In this work, we first establish a new framework, named Hierarchical ODL (HODL), to simultaneously investigate the intrinsic behaviors of optimization-derived model construction and its corresponding learning process. Then we rigorously prove the joint convergence of these two sub-tasks, from the perspectives of both approximation quality and stationary analysis. To our best knowledge, this is the first theoretical guarantee for these two coupled ODL components: optimization and learning. We further demonstrate the flexibility of our framework by applying HODL to challenging learning tasks, which have not been properly addressed by existing ODL methods. Finally, we conduct extensive experiments on both synthetic data and real applications in vision and other learning tasks to verify the theoretical properties and practical performance of HODL in various application scenarios.

**Index Terms**—Optimization-derived learning, meta optimization, hierarchical convergence analysis, constrained and regularized learning applications, bilevel optimization.

## 1 INTRODUCTION

Optimization-Derived Learning (ODL) is a class of methods for constructing deep models based on optimization techniques [1], [2] and has been widely used in different vision tasks in the past years [3], [4], [5], [6], [7]. Specifically, each of the optimization iteration is regarded as a layer of the network. All of these layers are concatenated to form a deep model. Passing through the network is equivalent to performing a finite number of optimization iterations. In addition, the optimization algorithm parameters (e.g., model parameters and regularization coefficients) are transferred to the learning variables in the network. In this way, the training network can be naturally interpreted as a parameterized optimization model, effectively overcoming the lack of interpretability in most of the traditional neural networks and leading to excellent performance as well.

Hence, a core problem of ODL is how to design the network structure based on the optimization model. In other words, ODL focuses on how to embed learnable modules into the optimization model. Depending on the way in which the learnable module is handled, existing approaches can be broadly classified into two main categories, respectively called ODL based on Unrolling with Numerical Hyper-

parameters (UNH), which aims to embed learnable modules under reliable theories and focuses on the convergence guarantee of the algorithm [8], [9], [10], and ODL Embedded with Network Architectures (ENA), which heuristically embeds learnable modules into the optimization algorithm, focusing more on the performance of practical tasks [11], [12], [13], [14], [15], [16]. Unfortunately, UNH usually treats optimization and networks as two separated modules, and ENA ignores the optimization process after designing the network structure. As a new framework, our Hierarchical ODL (HODL) also treats optimization and networks as two modules. However, unlike existing approaches, HODL establishes the nested relationship between optimization and networks via a hierarchical structure, and specifies the influence of learnable networks on the optimization process.

### 1.1 Related Works

As introduced in the last paragraph, existing ODL approaches are classified into UNH and ENA. Earlier UNH methods usually set learnable modules as some hyperparameters in the optimization algorithm which do not affect the convergence, such as the step size [8]. These methods avoid damaging the convergence results, but the number of learnable parameters is limited, making it hard to be applied to various practical tasks flexibly. In recent years, some UNH methods have embedded learnable modules to replace the descent direction in optimization as a novel perspective [9], [10]. In particular, they use the learnable module to provide the actual descent directions and set a convergence criterion to adjust it. For these methods, if the learnable modules are decoupled from the optimization model, the system flexibility is greatly enhanced. As for ENA, it often considers the optimization objective as a prototype to motivate its network model design for specific tasks. Specifically, ENA greatly improves the flexibility of traditional optimization by replacing some structures in the optimization model

- Risheng Liu and Xuan Liu are with the DUT-RU International School of Information Science and Engineering, Dalian University of Technology, Dalian 116024, China, and also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116024, China (e-mail: rslu@dlut.edu.cn, liuxuan\_16@126.com).
- Shangzhi Zeng is with the Department of Mathematics and Statistics, University of Victoria, Victoria, BC V8P5C2, Canada (e-mail: zengshangzhi@uvic.ca).
- Jin Zhang is with the Department of Mathematics, SUSTech International Center for Mathematics, Southern University of Science and Technology, National Center for Applied Mathematics Shenzhen, and Peng Cheng Laboratory, Shenzhen 518055, China (Corresponding author, e-mail: zhangj9@sustech.edu.cn).
- Yixuan Zhang is with the Department of Applied Mathematics, the Hong Kong Polytechnic University, Hong Kong SAR, China (e-mail: yi-xuan.zhang@connect.polyu.hk).

Manuscript received April 19, 2005; revised August 26, 2015.

TABLE 1: Various ODL methods whose base models include Proximal Gradient (PG), Augmented Lagrangian Method (ALM), and Half-Quadratic Splitting (HQS). Existing ODL methods are widely used in many fields, but the analysis of convergence, especially the convergence of learning variables  $\omega$ , is still insufficient.

Method	Category	Base Model	$\mathbf{u}^K \rightarrow \mathbf{u}^*$	$\inf \varphi_K(\omega) \rightarrow \inf \varphi(\omega)$	$\nabla \varphi(\omega) \rightarrow 0$	Application
ISTA-Net [17]	ENA	PG	✗	✗	✗	CS Reconstruction
ADMM-Net [18]		ALM	✗	✗	✗	CS-MRI
DUBLID [19]		HQS	✗	✗	✗	Image Deconvolution
LISTA [12]		PG	✓	✗	✗	Sparse Coding
DLADMM [13]		ALM	✓	✗	✗	Image Deblurring
PnP [20]		ALM, PG	✓	✗	✗	Image Super Resolution
PADNet [9]	UNH	ALM	✓	✗	✗	Image Haze Removal
FIMA [10]		PG	✓	✗	✗	Image Restoration
OISTA [8]		PG	✓	✗	✗	Sparse coding
HODL (Ours)	Both	Flexible	✓	✓	✓	Sparse Coding Image Restoration Hyper-parameter Optimization Few-shot Learning Generative Adversarial Learning

directly with learnable modules having similar effects. Some ENA methods only regard the linear layers of networks as learnable matrices. In LISTA [11] and CPSS [12], a non-linear feed-forward predictor is trained to produce the best approximation of sparse coding; in DLADMM [13], some learnable network modules are embedded into LADMM, and some learnable parameters are embedded into the proximal operator. In addition, some other ENA methods utilize more general networks. For instance, ISTA-Net [17] is based on ISTA as the fundamental iteration scheme, and adds a range of filters to learn parameters for image compressive sensing (CS); Plug-and-Play ADMM [14] replaces the projection gradient operator in ADMM with an implicit denoising module; pre-trained-CNN-based modules such as DPSR and DPIR [15], [16] are introduced to handle image restoration problems such as deconvolution, denoising, and super-resolution. Furthermore, a class of methods called learning to optimize [2], [21] can be regarded as ODL methods and unified under the HODL framework. Indeed, any method that utilizes an optimization model as the assistance in network construction falls within the scope of HODL. Specifically, approaches that focus on using the network to assist in optimizing the original objective function are classified into UNH, while those methods that utilize optimization as an assistance in network construction and aim to enhance the task performance rather than to optimize a specific objective function are referred to as ENA.

However, these existing ODL methods ignore the relationship between learnable modules and optimization models, leading to some drawbacks in methodology and theory. From the methodological perspective, since the convergence of UNH depends entirely on the original optimization algorithm, its performance is limited to manually designed target features, and it is impossible to further narrow the gap between target features and real-world tasks. ENA relies on the fact that the pre-trained network modules need to indeed have similar performance to the replaced part, which usually can only be promised by proper pre-training. Furthermore, existing ODL approaches have another common shortcoming in methodology: they deal with the optimization model and the learnable module separately, meaning that existing learnable modules are often trained independently of the

optimization model. While it is still possible to obtain the modules needed to optimize the model on the macroscopic level (e.g., replacing soft threshold operation with noise reduction modules), this has led to a gap between the modules needed to optimize the model and those that are actually learning. Although new methods exist to better isolate the learnable modules, this gap cannot be fundamentally addressed.

In terms of theoretical perspective, some works have analyzed the convergence of optimization process with the help of classic optimization techniques. To be specific, in [14], [22], [23] authors consider the non-expansive property of optimization iterative process under the condition that the embedded networks are bounded; in [20] the convergence is achieved when the Lipschitz constant of network residuals is strictly smaller than one. However, these works only focus on the convergence towards the fixed points of the approximated optimization model, but not the solution to the intrinsic task considering both optimization models and learnable modules. An intuitive treatment to handle this problem is to learn fewer learning variables. For example, in [8], only the step size of ISTA is learned, which nevertheless restricts the model. In addition, for ODL, additional artificially designed corrections are needed when learning the network. For example, in [10], [24], [25] authors manually design various rules to decide updates from the temporary updates generated by networks and optimization algorithms. However, the lack of learning variables and the manual design of rules severely limit these methods. Furthermore, ignoring the convergence of learning process also leads to some theoretical defects. First, as aforementioned, learning variables are fixed in the optimization process, and thus it is only able to consider the convergence of optimization variables, instead of the convergence of learning variables in learnable modules. Second, the learnable modules for ODL are too complex to determine the relationship between the true solution to the task and the obtained fixed points. Moreover, the convergence analysis of most existing ODL methods is developed from a specific optimization framework, so it is difficult to be extended to other optimization models.

## 1.2 Our Contributions

To address the aforementioned problems, we explicitly model ODL as a hierarchical relationship paradigm between the learnable module and the optimization algorithm, called HODL. Subsequently, in order to jointly train the optimization variables and the learning variables, we propose the corresponding solution strategy for solving HODL. We further put forward its simplified version to speed up the algorithm, and the simplified solution strategy can contain existing gradient-based unrolling algorithms as special cases. After that we provide the convergence analysis for this algorithmic framework. To be specific, we strictly prove the detailed theoretical properties to guarantee the joint convergence of optimization variables and learning variables, containing the convergence on approximation quality analysis and stationary analysis. We also conduct plenty of experiments on various learning and vision tasks to verify the effectiveness and wide applications of HODL. Our contributions can be summarized as follows, and the overall comparison of our HODL and existing ODL methods is displayed in Table 1.

- Unlike existing works that only pay attention to either learning or optimization process in ODL, we take both learning and optimization into consideration as two nested solution processes and formulate the general ODL paradigm, allowing us to further analyze the hierarchical relationship between the optimization and learning variables.
- From the hierarchical perspective, we build up the HODL framework and provide the novel and general ODL solution strategy. Our framework considers the nested relationship between optimization and learning, making it possible to jointly train optimization variables and learning variables.
- This work provides the strict joint convergence analysis of optimization variables and learning variables under the HODL framework, both on the approximation quality and on the stationary convergence. We additionally put forward a fast algorithm for HODL and its convergence analysis, which significantly extend the results in [3].
- We apply our HODL framework and the solution strategies to various learning tasks, containing sparse coding as the toy example, and image processing tasks (e.g., rain streak removal, image deconvolution, and low-light enhancement). In addition, our HODL can also handle bilevel optimization tasks that cannot be handled by existing ODL methods, such as adversarial learning, hyper-parameter optimization, and few-shot learning.

## 2 THE PROPOSED ALGORITHMIC FRAMEWORK

In this section, we first put forward the general ODL paradigm, and introduce our Hierarchical Optimization-Derived Learning (HODL) framework to unify the optimization algorithms and learnable modules. Then the solution strategies for this HODL framework are provided.

### 2.1 The General ODL Paradigm

ODL usually translates the application problem into two parts of the optimization problem, the task term and the learn-

able term, with respect to the optimization variable  $\mathbf{u} \in U$ . The task term is usually an objective function  $f(\mathbf{u})$  that represents the dependence of the solution of  $\mathbf{u}$  on the task itself. The learnable term, on the other hand, can be classified into two common forms, the regularization term  $g(\mathbf{u})$  and the linear constraint term  $\mathcal{A}(\mathbf{u}) = \mathbf{y}$ , which are used to represent the task prior that aids in solving the problem. Hence, ODL usually transforms the specific task into the following form

$$\min_{\mathbf{u} \in U} \underbrace{f(\mathbf{u})}_{\text{Task Term}} + \underbrace{g(\mathbf{u}, \boldsymbol{\omega})}_{\text{Regularization}} \text{ and/or } \underbrace{\text{s.t. } \mathcal{A}(\mathbf{u}, \boldsymbol{\omega}) = \mathbf{y}(\boldsymbol{\omega})}_{\text{Learnable Term Constraint}}, \quad (1)$$

where  $\boldsymbol{\omega}$ , the parameters of learnable term, is called the learning variable. Denote the solution set with respect to  $\mathbf{u}$  for a given  $\boldsymbol{\omega}$  to be  $\mathcal{S}(\boldsymbol{\omega})$ , and denote the corresponding algorithmic operator for solving Eq. (1) to be  $\mathcal{D}$ . In classical optimization methods  $\mathcal{D}$  is usually constructed manually by optimization experts based on theory and experience. As a paradigm for designing network structures, ODL designs the network from an optimization perspective. To be specific, by building the model based on classical optimization process as the structural basis and embedding learnable modules, ODL generates a complete network structure with both interpretability of optimization models and learnability of neural networks. This paradigm is flexible enough that the learnable module can be not only the hyper-parameters in the numerical optimization process, but also the entire networks used to replace certain process steps. The corresponding networks are respectively denoted as  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$ .

Unfortunately, existing ODL methods only consider optimization when building the initial network structure, and follow the ordinary deep neural network strategy during training, instead of combining optimization and learning. This splits ODL into two parts: during the training procedure, they only care about the convergence of learning variables  $\boldsymbol{\omega}$  and ignore the iterations of optimization variables  $\mathbf{u}$ ; while in testing, they fix  $\boldsymbol{\omega}$  and hope  $\mathbf{u}$  to converge in the optimization process under the fixed network structure.

### 2.2 Our Meta Optimization Framework<sup>1</sup>

To address the fragmentation of optimization and learning processes, we use the idea of optimization not only when building the network structure, but also during the training procedure. Despite the embedded learnable module, the network structure of ODL can still be considered as an optimization process for solving a specific problem. Hence, by nesting the results of the optimization process into the inputs of the learning process, for the problem in Eq. (1), we can transfer it to the following

$$\min_{\mathbf{u} \in U, \boldsymbol{\omega} \in \Omega} \ell(\mathbf{u}, \boldsymbol{\omega}), \text{ s.t. } \mathbf{u} \in \mathcal{S}(\boldsymbol{\omega}), \quad (2)$$

where  $\ell$  is the objective function.

Next, we put forward a unified form in dealing with all kinds of problems in Eq. (1), which also facilitates our subsequent analysis. Specifically, each iteration of the ODL method constitutes an operator origin from the optimization

1. In numerical optimization, meta-optimization is the use of one optimization method to tune another optimization method [26].

algorithm but embedded with a learnable module, and the result of a stable iteration is taken as the output of ODL. Therefore, a reasonable assumption is to consider the operator as non-expansive and the output of ODL as the fixed point of the corresponding iterative operator for solving Eq. (1). Therefore, we model the optimal solution of ODL uniformly by  $\mathbf{u} = \mathcal{D}(\mathbf{u}, \omega)$  to find the fixed point, where  $\omega$  is the learning variable, and  $\mathcal{D}$  is the non-expansive operator. Here  $\mathcal{D}(\cdot, \omega) \in \{\mathcal{D}_{\text{num}}(\cdot, \omega) \circ \mathcal{D}_{\text{net}}(\cdot, \omega)\}$ , where  $\circ$  represents compositions of operators. Same as introduced in Section 2.1,  $\mathcal{D}_{\text{num}}$  regards the hyper-parameters in the numerical optimization process as learnable modules, while  $\mathcal{D}_{\text{net}}$  replaces certain process steps to be networks directly. Hence, this form not only includes optimization algorithms, but also contains other implicitly defined models, which originate from optimization but are added with learnable modules additionally. The process to find the fixed point can be implemented via the classical Krasnoselskii-Mann updating scheme [27] generalized with learning variables  $\omega$ , in the form of  $\mathcal{T}(\mathbf{u}^k, \omega) = \mathbf{u}^k + \alpha(\mathcal{D}(\mathbf{u}^k, \omega) - \mathbf{u}^k)$ , as the  $k$ -th iteration step, where  $\alpha \in (0, 1)$ . Note that if  $\mathcal{D}$  is non-expansive, then  $\mathcal{T}$  is an  $\alpha$ -averaged non-expansive operator. Furthermore, the fixed point of  $\mathcal{D}$  is also a fixed point of  $\mathcal{T}$ . In experiments, for guaranteeing that  $\mathcal{D}$  is non-expansive, some normalization techniques such as spectral normalization [28] are implemented on parameters. By choosing the solution of the fixed point problem  $\mathbf{u} = \mathcal{T}(\mathbf{u}, \omega)$  as the input for learning  $\omega$ , the hierarchical formulation of a general ODL problem can be expressed in the following form

$$\min_{\mathbf{u} \in U, \omega \in \Omega} \ell(\mathbf{u}, \omega), \text{ s.t. } \mathbf{u} = \mathcal{T}(\mathbf{u}, \omega), \quad (3)$$

where  $\ell$  is the loss function corresponding to the learning process, and  $\mathcal{T}$  denotes the optimization process. We call problems of this formulation as Hierarchical Optimization-Derived Learning (HODL), which also serve as our meta optimization framework. In learning to optimize, a class of methods are designed to learn an optimizer to optimize an objective function for a specific task, and this optimizer corresponds to the operator  $\mathcal{T}$  under our HODL framework.

Actually, HODL can overcome several shortcomings in existing ODL methods mentioned in Section 1 thanks to its hierarchical modeling. From the viewpoint of theory, HODL makes it possible to study the joint convergence of  $\omega$  and  $\mathbf{u}$  under their nested relationship, in place of only considering one of them independently. Hence, instead of only obtaining the fixed points of the optimization process for a fixed  $\omega$ , we can approach the true optimal solution of the whole problem. We will provide the detailed convergence analysis in Section 3. From the viewpoint of applications, in the practical training procedure, the learning variables  $\omega$  are also adjusted along with the iterations of optimization variables  $\mathbf{u}$ , rather than just embedding a network that ignores the optimization structure.

### 2.3 Efficient Solution Strategy

Next we establish the algorithm to simultaneously solve the optimization variables  $\mathbf{u}$  and learning variables  $\omega$ . Existing ODL methods usually update the optimization variables with fixed pre-trained learning variables, ignoring the nested rela-

tionships in ODL when training the optimization variables and learning variables and failing to solve them together.

**The Nested Learning Iteration.** To begin with, the learning variables  $\omega$  are nested into the optimization variables  $\mathbf{u}$ . Note that existing ODL approaches ignore the hierarchical structure of  $\omega$  and  $\mathbf{u}$  in modeling, so their algorithms also do not contain their hierarchy and are unavailable under our HODL framework. We design the training of  $\omega$  in order that the nested relationship between  $\mathbf{u}$  and  $\omega$  can be effectively exploited. Specifically, each iterative step of  $\mathbf{u}$  is parameterized by  $\omega$ , so the iteration result of  $\mathbf{u}$  is a function of  $\omega$ , i.e.,  $\mathbf{u}^k(\omega)$ . This reveals the dependence of optimization variables  $\mathbf{u}$  on the learning variables  $\omega$ , and thus the complete optimization iteration of  $\mathbf{u}$  (inner loop) is embedded within the learning iteration of  $\omega$  (outer loop). Hence, the objective function of learning  $\omega$  contains the entire iterative trajectory of  $\mathbf{u}$ , which effectively exploits their nested relationship.

**The Nested Optimization Iteration.** For the iteration of optimization variables  $\mathbf{u}$ , we also add an additional nested structure related to the learning process. To begin with, we compute the iterative direction  $\mathbf{v}_l$  from the optimization process corresponding to  $\mathcal{T}$  in Eq. (3) (lower level). At the  $k$ -th step, to approach the fixed point of  $\mathcal{T}(\cdot, \omega)$  for a given  $\omega$ ,  $\mathbf{v}_l^k = \mathcal{T}(\mathbf{u}^{k-1}, \omega)$  is defined as an update direction of  $\mathbf{u}$ . Note that here the operator  $\mathcal{T}$  is adjusted to be non-expansive under the induced norm  $\|\cdot\|_{\mathbf{G}_\omega}$  where  $\mathbf{G}_\omega$  is a positive-definite correction matrix parameterized by  $\omega$  and will be discussed in detail in Section 3. Next, we compute another iterative direction  $\mathbf{v}_u$  from the learning process in Eq. (3) (upper level). It makes our updating direction of  $\mathbf{u}$  able to utilize the information of  $\omega$  by using the gradient of loss function  $\ell$  with respect to  $\mathbf{u}$ . Nevertheless, directly applying its gradient may destroy the non-expansive property with respect to  $\|\cdot\|_{\mathbf{G}_\omega}$ . Consequently, for the consistent non-expansive property with direction  $\mathbf{v}_l$ , we further add an additional correction  $\mathbf{G}_\omega^{-1}$  to the gradient of  $\ell$ , and request the corresponding step sizes  $s_k$  to be a decreasing sequence for assuring the correctness of this iterative direction  $\mathbf{v}_u$ , i.e.,  $\mathbf{v}_u^k = \mathbf{u}^{k-1} - s_k \mathbf{G}_\omega^{-1} \frac{\partial}{\partial \mathbf{u}} \ell(\mathbf{u}^{k-1}, \omega)$ , where  $s_k \rightarrow 0$  as  $k$  increases. Lastly, inspired by [29], we generate the final updating direction of  $\mathbf{u}$  by aggregating the two iterative directions  $\mathbf{v}_l$  and  $\mathbf{v}_u$  via a linear combination under the projection, i.e.,  $\mathbf{u}^k = \text{Proj}_{U, \mathbf{G}_\omega}(\mu \mathbf{v}_u^k + (1 - \mu) \mathbf{v}_l^k)$ , where  $\mu \in (0, 1)$ . Here the projection operator  $\text{Proj}_{U, \mathbf{G}_\omega}(\cdot)$  is associated to  $\mathbf{G}_\omega$  with the definition  $\text{Proj}_{U, \mathbf{G}_\omega}(\mathbf{u}) = \arg\min_{\bar{\mathbf{u}} \in U} \|\bar{\mathbf{u}} - \mathbf{u}\|_{\mathbf{G}_\omega}$ . Note that in the theoretical analysis part, the projection is only used to guarantee the boundedness of  $\mathbf{u}^k$ ; while in practical experiments and applications, generally  $U$  is set to be such a large bounded set or even unbounded  $\mathbb{R}^n$  that the projection operator can be ignored. To conclude, the iterations of optimization variables  $\mathbf{u}$  in our solution strategy for HODL reads as

$$\begin{cases} \mathbf{v}_l^k(\omega) = \mathcal{T}(\mathbf{u}^{k-1}(\omega), \omega), \\ \mathbf{v}_u^k(\omega) = \mathbf{u}^{k-1}(\omega) - s_k \mathbf{G}_\omega^{-1} \frac{\partial}{\partial \mathbf{u}} \ell(\mathbf{u}^{k-1}(\omega), \omega), \\ \mathbf{u}^k(\omega) = \text{Proj}_{U, \mathbf{G}_\omega}(\mu \mathbf{v}_u^k(\omega) + (1 - \mu) \mathbf{v}_l^k(\omega)), \end{cases} \quad (4)$$

where  $k = 1, \dots, K$ .

Here our solution strategy to solve the HODL problem is with aggregation of  $\mathbf{v}_l$  and  $\mathbf{v}_u$ , so it is shortened as HODL



with aggregation (aHODL for short). On the other hand, from the viewpoint of computational efficiency in practical applications, the algorithm can be further improved. To be specific, a computational drawback comes from the need for gradual decay of  $s_k$  in Eq. (4), which leads to an increase in the number of training iteration. In addition,  $\mathbf{G}_\omega^{-1}$  may be challenging to compute according to different forms of  $\mathcal{D}$ , and even  $\mathbf{G}_\omega$  itself may be hard to estimate. Therefore, we adjust aHODL and put forward a simplified HODL (sHODL for short) without the aggregation step as in Eq. (4). That is, we let  $\mu$  in Eq. (4) to be 0, and then  $\mathbf{u}^K(\omega)$  is iterated as

$$\mathbf{u}^k(\omega) = \text{Proj}_{U, \mathbf{G}_\omega} \left( \mathbf{v}_l^k(\omega) \right), \quad (5)$$

where  $\mathbf{v}_l^k(\omega) = \mathcal{T}(\mathbf{u}^{k-1}(\omega), \omega)$ , and  $k = 1, \dots, K$ . Compared with Eq. (4), sHODL, the strategy without aggregation, is simpler to implement with higher efficiency as a fast algorithm than aHODL. Hence, our HODL framework can then be extended to more application tasks. Convergence of aHODL and sHODL will be discussed in the next section, which also indicates the superiority of aHODL over sHODL in theory. The algorithmic flow for aHODL and sHODL is summarized in Algorithm 1.

---

**Algorithm 1** HODL

---

**Require:** Step sizes  $\{s_k\}$ ,  $\gamma$  and parameter  $\mu$ .

- 1: Initialize  $\omega^0$ .
  - 2: **for**  $t = 1 \rightarrow T$  **do**
  - 3:   Initialize  $\mathbf{u}^0$ .
  - 4:   **for**  $k = 1 \rightarrow K$  **do**
  - 5:     Compute  $\mathbf{u}^k$  by Eq. (4) (aHODL) or the simplified version Eq. (5) (sHODL).
  - 6:   **end for**
  - 7:    $\omega^t = \text{Proj}_{\Omega, \mathbf{G}_\omega} (\omega^{t-1} - \gamma \frac{\partial}{\partial \omega} \ell(\mathbf{u}^K(\omega^{t-1}), \omega^{t-1}))$ .
  - 8: **end for**
- 

### 3 THEORETICAL ANALYSIS

In this section, we propose the convergence analysis of the solution strategies for HODL problems in Eq. (3) with respect to both optimization variables  $\mathbf{u}$  and learning variables  $\omega$ . Our analysis for the solution strategy of HODL is separated into two parts, the approximation quality analysis on the convergence of optimal value in Section 3.1, and the stationary analysis on the convergence of stationary points in Section 3.2. For the simplified solution strategy without aggregation sHODL as mentioned in Section 2.3, we also provide further analysis in Section 3.3. Note that since HODL in Eq. (3) is a general form of ODL problems, our analysis also serves as a unified route of theoretical analysis for other methods and more problems with hierarchical structures.

To begin with, we denote the fixed point set of operator  $\mathcal{T}$  to be  $\text{Fix}(\mathcal{T}(\cdot, \omega))$  for a given  $\omega$ , and then the HODL problem in Eq. (3) can be rewritten as

$$\min_{\omega \in \Omega} \varphi(\omega), \quad \text{where} \quad \varphi(\omega) := \inf_{\mathbf{u} \in \text{Fix}(\mathcal{T}(\cdot, \omega)) \cap U} \ell(\mathbf{u}, \omega). \quad (6)$$

In Algorithm 1,  $\mathbf{u}^K(\omega)$  is obtained by iterating as Eq. (4) (aHODL) or its simplification in Eq. (5) (sHODL), to solve

the simple bilevel problem  $\inf_{\mathbf{u} \in \text{Fix}(\mathcal{T}(\cdot, \omega)) \cap U} \ell(\mathbf{u}, \omega)$ . Substituting  $\mathbf{u}^K(\omega)$  for  $\mathbf{u}$  in  $\ell(\mathbf{u}, \omega)$  of Eq. 6, we have its approximation problem as the following

$$\min_{\omega \in \Omega} \varphi_K(\omega) := \ell(\mathbf{u}^K(\omega), \omega), \quad (7)$$

which is only about the variable  $\omega$ , and is solved by the sequence  $\{\omega^t\}$  generated by Algorithm 1.

#### 3.1 Approximation Quality Analysis

In this part, we show that Eq. (7) obtained by aHODL is actually an appropriate approximation to Eq. (3), meaning that any limit point  $(\bar{\mathbf{u}}, \bar{\omega})$  of the sequence  $\{(\mathbf{u}^K(\omega^K), \omega^K)\}$  is a solution to the HODL problem in Eq. (3), where  $\omega^K \in \arg\min_{\omega \in \Omega} \varphi_K(\omega)$  as a solution to Eq. (7) is generated by Algorithm 1 and  $\mathbf{u}^K(\omega)$  is computed from Eq. (4). Hence, we can approach the optimal solution of HODL in Eq. (3) by solving Eq. (7).

We make the following standing assumptions throughout this part, and then show that Algorithm 1 can achieve convergence in the sense of approximation quality under mild conditions.

**Assumption 3.1**  $\Omega$  is a compact set and  $U$  is a convex compact set.  $\text{Fix}(\mathcal{T}(\cdot, \omega))$  is nonempty for any  $\omega \in \Omega$ .  $\ell(\mathbf{u}, \omega)$  is continuous on  $\mathbb{R}^n \times \Omega$ . For any  $\omega \in \Omega$ ,  $\ell(\cdot, \omega) : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $L_\ell$ -smooth, convex and bounded below by  $M_0$ .

Please notice that function  $\ell$  is usually defined to be the MSE loss, so Assumption 3.1 is quite standard for ODL problems [20], [16]. Next we present some necessary preliminaries. For any two matrices  $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{R}^{n \times n}$ , we consider the following partial ordering relation:

$$\mathbf{G}_1 \succeq \mathbf{G}_2 \Leftrightarrow \langle \mathbf{u}, \mathbf{G}_1 \mathbf{u} \rangle \geq \langle \mathbf{u}, \mathbf{G}_2 \mathbf{u} \rangle, \quad \forall \mathbf{u} \in \mathbb{R}^n.$$

If  $\mathbf{G} \succ 0$ , then  $\langle \mathbf{u}_1, \mathbf{G} \mathbf{u}_2 \rangle$  for  $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^n$  defines an inner product on  $\mathbb{R}^n$ . Denote the induced norm with  $\|\cdot\|_{\mathbf{G}}$ , i.e.,  $\|\mathbf{u}\|_{\mathbf{G}} := \sqrt{\langle \mathbf{u}, \mathbf{G} \mathbf{u} \rangle}$  for any  $\mathbf{u} \in \mathbb{R}^n$ . We assume that  $\mathcal{D}(\cdot, \omega)$  satisfies the following assumptions throughout this part.

**Assumption 3.2** There exist  $\mathbf{G}_{ub} \succeq \mathbf{G}_{lb} \succ 0$ , such that for each  $\omega \in \Omega$ , there exists  $\mathbf{G}_{ub} \succeq \mathbf{G}_\omega \succeq \mathbf{G}_{lb}$  such that

- (1)  $\mathcal{D}(\cdot, \omega)$  is non-expansive with respect to  $\|\cdot\|_{\mathbf{G}_\omega}$ , i.e., for all  $(\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{R}^n \times \mathbb{R}^n$ ,

$$\|\mathcal{D}(\mathbf{u}_1, \omega) - \mathcal{D}(\mathbf{u}_2, \omega)\|_{\mathbf{G}_\omega} \leq \|\mathbf{u}_1 - \mathbf{u}_2\|_{\mathbf{G}_\omega}.$$

- (2)  $\mathcal{D}(\cdot, \omega)$  is closed, i.e.,  $\text{gph } \mathcal{D}(\cdot, \omega)$  is closed, where

$$\text{gph } \mathcal{D}(\cdot, \omega) := \{(\mathbf{u}, \mathbf{v}) \in \mathbb{R}^n \times \mathbb{R}^n \mid \mathbf{v} = \mathcal{D}(\mathbf{u}, \omega)\}.$$

The non-expansive property of  $\mathcal{T}(\cdot, \omega)$  in Eq. (3) can be obtained immediately from that of  $\mathcal{D}(\cdot, \omega)$  in Assumption 3.2 [30][Proposition 4.25]. Then we can prove that the sequence  $\{\mathbf{u}^k(\omega)\}$  generated by Eq. (4) not only converges to the solution set of  $\inf_{\mathbf{u} \in \text{Fix}(\mathcal{T}(\cdot, \omega)) \cap U} \ell(\mathbf{u}, \omega)$ , but also admits a uniform convergence towards the fixed point set  $\text{Fix}(\mathcal{T}(\cdot, \omega))$  with respect to  $\|\mathbf{u}^k(\omega) - \mathcal{T}(\mathbf{u}^k(\omega), \omega)\|_{\mathbf{G}_{lb}}^2$  for  $\omega \in \Omega$ . Thanks to the uniform convergence property of the sequence  $\{\mathbf{u}^k(\omega)\}$ , inspired by the arguments used in [29], we can establish the convergence on both  $\mathbf{u}$  and  $\omega$  of Algorithm 1 towards the solution of HODL problem

in Eq. (3). The convergence results of approximation quality are summarized in the following theorem. Please refer to our conference version in [3] for detailed proofs.

**Theorem 3.1** Suppose Assumptions 3.1 and 3.2 are satisfied. Let  $\{\mathbf{u}^k(\boldsymbol{\omega})\}$  be the sequence generated by Eq. (4) with  $\mu \in (0, 1)$  and  $s_k = \frac{s}{k+1}$ , where  $s \in (0, \frac{\lambda_{\min}(\mathbf{G}_{lb})}{L_\ell})$ , and  $\lambda_{\min}(\mathbf{G}_{lb})$  denotes the smallest eigenvalue of matrix  $\mathbf{G}_{lb}$ .

(1) For any  $\boldsymbol{\omega} \in \Omega$ , we have

$$\lim_{k \rightarrow \infty} \text{dist}(\mathbf{u}^k(\boldsymbol{\omega}), \text{Fix}(\mathcal{T}(\cdot, \boldsymbol{\omega}))) = 0,$$

and

$$\lim_{k \rightarrow \infty} \ell(\mathbf{u}^k(\boldsymbol{\omega}), \boldsymbol{\omega}) = \varphi(\boldsymbol{\omega}).$$

Furthermore, there exists  $C > 0$  such that for any  $\boldsymbol{\omega} \in \Omega$ ,

$$\|\mathbf{u}^k(\boldsymbol{\omega}) - \mathcal{T}(\mathbf{u}^k(\boldsymbol{\omega}), \boldsymbol{\omega})\|_{\mathbf{G}_{lb}}^2 \leq C \sqrt{\frac{1 + \ln(1+k)}{k^{\frac{1}{4}}}}.$$

(2) Let  $\boldsymbol{\omega}^K \in \arg\min_{\boldsymbol{\omega} \in \Omega} \varphi_K(\boldsymbol{\omega})$ , and we have any limit point  $(\bar{\mathbf{u}}, \bar{\boldsymbol{\omega}})$  of the sequence  $\{(\mathbf{u}^K(\boldsymbol{\omega}^K), \boldsymbol{\omega}^K)\}$  is a solution to the problem in Eq. (3), i.e.,  $\bar{\boldsymbol{\omega}} \in \arg\min_{\boldsymbol{\omega} \in \Omega} \varphi(\boldsymbol{\omega})$  and  $\bar{\mathbf{u}} = \mathcal{T}(\bar{\mathbf{u}}, \bar{\boldsymbol{\omega}})$ . Furthermore,  $\inf_{\boldsymbol{\omega} \in \Omega} \varphi_K(\boldsymbol{\omega}) \rightarrow \inf_{\boldsymbol{\omega} \in \Omega} \varphi(\boldsymbol{\omega})$  as  $K \rightarrow \infty$ .

### 3.2 Stationary Analysis

Next, we put forward the convergence analysis of our solution strategy with aggregation aHODL (using Eq. (4) to compute  $\mathbf{u}^K$  in Algorithm 1) on stationary points. That is, for any limit point  $\bar{\boldsymbol{\omega}}$  of the sequence  $\{\boldsymbol{\omega}^K\}$ , we have  $\nabla \varphi(\bar{\boldsymbol{\omega}}) = 0$ , where  $\varphi(\boldsymbol{\omega})$  is defined in Eq. (6).

Here we make  $U = \mathbb{R}^n$  and suppose the operator  $\mathcal{T}$  has a unique fixed point, which means the fixed point set  $\text{Fix}(\mathcal{T}(\cdot, \boldsymbol{\omega}))$  is a singleton. We denote the unique solution by  $\mathbf{u}^*(\boldsymbol{\omega})$ . Our analysis is partly inspired by [29] and [31].

**Assumption 3.3**  $\Omega$  is a compact set and  $U = \mathbb{R}^n$ .  $\text{Fix}(\mathcal{T}(\cdot, \boldsymbol{\omega}))$  is nonempty for any  $\boldsymbol{\omega} \in \Omega$ .  $\ell(\mathbf{u}, \boldsymbol{\omega})$  is twice continuously differentiable on  $\mathbb{R}^n \times \Omega$ . For any  $\boldsymbol{\omega} \in \Omega$ ,  $\ell(\cdot, \boldsymbol{\omega}) : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $L_\ell$ -smooth, convex and bounded below by  $M_0$ .

For  $\mathcal{D}(\cdot, \boldsymbol{\omega})$  we request a stronger assumption than Assumption 3.2 that  $\mathcal{D}(\cdot, \boldsymbol{\omega})$  is contractive with respect to  $\|\cdot\|_{\mathbf{G}_\omega}$  throughout this part, to guarantee the uniqueness of the fixed point.

**Assumption 3.4** There exist  $\mathbf{G}_{ub} \succeq \mathbf{G}_{lb} \succ 0$ , such that for each  $\boldsymbol{\omega} \in \Omega$ , there exists  $\mathbf{G}_{ub} \succeq \mathbf{G}_\omega \succeq \mathbf{G}_{lb}$  such that

(1)  $\mathcal{D}(\cdot, \boldsymbol{\omega})$  is contractive with respect to  $\|\cdot\|_{\mathbf{G}_\omega}$ , i.e., there exists  $\bar{\rho} \in (0, 1)$ , such that for all  $(\mathbf{u}_1, \mathbf{u}_2) \in \mathbb{R}^n \times \mathbb{R}^n$ ,

$$\|\mathcal{D}(\mathbf{u}_1, \boldsymbol{\omega}) - \mathcal{D}(\mathbf{u}_2, \boldsymbol{\omega})\|_{\mathbf{G}_\omega} \leq \bar{\rho} \|\mathbf{u}_1 - \mathbf{u}_2\|_{\mathbf{G}_\omega}.$$

(2)  $\mathcal{D}(\cdot, \boldsymbol{\omega})$  is closed.

Denote  $\hat{\mathcal{S}}(\boldsymbol{\omega}) := \arg\min_{\mathbf{u} \in \text{Fix}(\mathcal{T}(\cdot, \boldsymbol{\omega})) \cap U} \ell(\mathbf{u}, \boldsymbol{\omega})$ , and we have the following stationary analysis results.

**Theorem 3.2** Suppose Assumptions 3.3 and 3.4 are satisfied,  $\frac{\partial}{\partial \mathbf{u}} \mathcal{T}(\mathbf{u}, \boldsymbol{\omega})$  and  $\frac{\partial}{\partial \boldsymbol{\omega}} \mathcal{T}(\mathbf{u}, \boldsymbol{\omega})$  are Lipschitz continuous with respect to  $\mathbf{u}$ , and  $\hat{\mathcal{S}}(\boldsymbol{\omega})$  is nonempty for all  $\boldsymbol{\omega} \in \Omega$ . Let  $\{\mathbf{u}^k(\boldsymbol{\omega})\}$  be the

sequence generated by Eq. (4) with  $\mu \in (0, 1)$  and  $s_k = \frac{s}{k+1}$ , where  $s \in (0, \frac{\lambda_{\min}(\mathbf{G}_{lb})}{L_\ell})$ .

(1) We have

$$\sup_{\boldsymbol{\omega} \in \Omega} \|\nabla \varphi_k(\boldsymbol{\omega}) - \nabla \varphi(\boldsymbol{\omega})\|_{\mathbf{G}} \rightarrow 0, \text{ as } k \rightarrow \infty.$$

(2) Let  $\boldsymbol{\omega}^K$  be an  $\varepsilon_K$ -stationary point of  $\varphi_K(\boldsymbol{\omega})$ , i.e.,

$$\varepsilon_K = \nabla \varphi_K(\boldsymbol{\omega}^K).$$

Then if  $\varepsilon_K \rightarrow 0$ , we have that any limit point  $\bar{\boldsymbol{\omega}}$  of the sequence  $\{\boldsymbol{\omega}^K\}$  is a stationary point of  $\varphi$ , i.e.,

$$0 = \nabla \varphi(\bar{\boldsymbol{\omega}}).$$

For detailed proofs of the above results, please refer to our conference version in [3].

### 3.3 Convergence of HODL without Aggregation (sHODL)

In Section 3.1 and 3.2, we discuss the convergence properties (approximation quality and stationary analysis) of solution strategy aHODL (using Eq. (4) to compute  $\mathbf{u}^K$ ). Now we further extend these convergence properties to the solution strategy without aggregation sHODL introduced in Section 2.3 (using Eq. (5) to compute  $\mathbf{u}^K$ ).

On the approximation quality, based on Assumptions 3.1 and 3.2, under the further assumptions that  $\ell(\cdot, \boldsymbol{\omega})$  is uniformly Lipschitz continuous and  $\mathcal{T}(\cdot, \boldsymbol{\omega})$  has a unique fixed point, the approximation quality result for the solution strategy without aggregation can be obtained. For detailed discussions please refer to [32], [33]. Note that for the convergence guarantee, compared with aHODL, the simplified solution strategy sHODL reduces the computational burden but requires a stronger assumption that the operator  $\mathcal{T}$  is contractive, i.e., the set  $\text{Fix}(\mathcal{T}(\cdot, \boldsymbol{\omega}))$  is a singleton. Also note that in this situation the convexity of  $\ell$  is not required. Corresponding to those classic gradient-based unrolling algorithms without linear constraints, they require that the objective function in Eq. (1) is strongly convex [34], [32]. If the solutions to the optimization process are not unique (such as  $f$  is only convex, i.e., the corresponding operator is only non-expansive), and substituted to the learning process directly, then the obtained solution may be far away from the true solution of the original bilevel problem. Please refer to the counter-example in [33]. However, using the solution strategy with aggregation aHODL (using Eq. (4) to compute  $\mathbf{u}^K$ ) which aggregates the upper and lower iterative directions  $\mathbf{v}_l$  and  $\mathbf{v}_u$ , then even if the fixed points are not unique (the lower iterative operator is merely non-expansive), we can still approach the true solution with joint convergence.

On the stationary analysis, please note that our stationary analysis in Section 3.2 is also a unified convergence analysis of our solution strategies with and without aggregation (aHODL and sHODL), so it is applicable to all kinds of hierarchical problems. Specifically,  $\mu$  in aHODL (using Eq. (4) to compute  $\mathbf{u}^K$ ) is taken to be between 0 and 1, while in the solution strategy without aggregation sHODL (using the simplified form Eq. (5) to compute  $\mathbf{u}^K$ ), it is taken to be 0. Taking  $\mu = 0$ , Theorem 3.2 also holds, and the proofs parallel. Please also refer to [34] for the stationary analysis of the classic gradient-based unrolling algorithms as the

special case of our solution strategy without aggregation. The discussions above for the convergence properties of HODL without aggregation (sHODL) can be concluded in the following proposition.

**Proposition 3.1** Suppose  $\{\mathbf{u}^k(\boldsymbol{\omega})\}$  to be the sequence generated by sHODL in Section 2.3.

- (1) Suppose Assumptions 3.1 and 3.2 are satisfied,  $\ell(\cdot, \boldsymbol{\omega})$  is uniformly Lipschitz continuous and  $\mathcal{T}(\cdot, \boldsymbol{\omega})$  has a unique fixed point. Then, let  $\boldsymbol{\omega}^K \in \operatorname{argmin}_{\boldsymbol{\omega} \in \Omega} \varphi_K(\boldsymbol{\omega})$ , and we have any limit point  $(\bar{\mathbf{u}}, \bar{\boldsymbol{\omega}})$  of the sequence  $\{(\mathbf{u}^K(\boldsymbol{\omega}^K), \boldsymbol{\omega}^K)\}$  is a solution to the problem in Eq. (3), i.e.,  $\bar{\boldsymbol{\omega}} \in \operatorname{argmin}_{\boldsymbol{\omega} \in \Omega} \varphi(\boldsymbol{\omega})$  and  $\bar{\mathbf{u}} = \mathcal{T}(\bar{\mathbf{u}}, \bar{\boldsymbol{\omega}})$ . Further,  $\inf_{\boldsymbol{\omega} \in \Omega} \varphi_K(\boldsymbol{\omega}) \rightarrow \inf_{\boldsymbol{\omega} \in \Omega} \varphi(\boldsymbol{\omega})$  as  $K \rightarrow \infty$ .
- (2) Suppose Assumptions 3.3 and 3.4 are satisfied,  $\frac{\partial}{\partial \mathbf{u}} \mathcal{T}(\mathbf{u}, \boldsymbol{\omega})$  and  $\frac{\partial}{\partial \boldsymbol{\omega}} \mathcal{T}(\mathbf{u}, \boldsymbol{\omega})$  are Lipschitz continuous with respect to  $\mathbf{u}$ , and  $\hat{S}(\boldsymbol{\omega})$  is nonempty for all  $\boldsymbol{\omega} \in \Omega$ . Let  $\boldsymbol{\omega}^K$  be an  $\varepsilon_K$ -stationary point of  $\varphi_K(\boldsymbol{\omega})$ , i.e.,  $\varepsilon_K = \nabla \varphi_K(\boldsymbol{\omega}^K)$ . Then if  $\varepsilon_K \rightarrow 0$ , we have that any limit point  $\bar{\boldsymbol{\omega}}$  of the sequence  $\{\boldsymbol{\omega}^K\}$  is a stationary point of  $\varphi$ , i.e.,  $0 = \nabla \varphi(\bar{\boldsymbol{\omega}})$ .

## 4 APPLICATIONS

In this section, we first compare HODL with other established ODL methods in detail, and then demonstrate the applications of HODL in solving practical problems of various forms and the specific settings under these forms. Summary of operators  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$  for problems of various forms and corresponding applications is shown in Table 2, where the applications for other learning tasks regarded as hierarchical models will be discussed in Section 5.

### 4.1 Comparison with Existing ODL Methods

Compared with existing ODL methods, HODL additionally considers the optimal update of learning variables  $\boldsymbol{\omega}$ , thus providing better theoretical guarantees and higher application value. Existing ODL methods only focus on the output of optimization model, i.e., the final iterative results of the optimization variables  $\mathbf{u}$ . Usually, their selection of learning variables  $\boldsymbol{\omega}$  is just a direct extraction of network modules from similar learning tasks [17], [18]. Hence, this selection method ignores the convergence of learning variables  $\boldsymbol{\omega}$  and can be considered as the optimization strategy of random search for similar learning tasks in the search space of learning variables  $\boldsymbol{\omega}$ . On the contrary, HODL focuses on the iterative results of both optimization variables  $\mathbf{u}$  and learning variables  $\boldsymbol{\omega}$ , and performs gradient descent on learning variables  $\boldsymbol{\omega}$ , thus providing sufficient theoretical guarantees and clear application framework. In a word, compared with existing ODL methods, HODL makes up for the weakness of ODL in theory and upgrades from random search to gradient descent for application, providing the theoretical guarantee and usability of ODL models that existing methods cannot achieve. Under the HODL framework, the difference among algorithms for various applications lies in the operator  $\mathcal{D}$  introduced in Section 2.2. Next we introduce the specific forms of  $\mathcal{D}$  in these applications.

### 4.2 Application for Sparse Coding

Taking sparse coding as an example, we first describe how HODL can be applied to constrained and regularized problems and show how the coupling between the optimization model and optimization variables can be handled. Specifically, the sparse coding task is dedicated to representing given data  $\mathbf{b}$  as a sparse coefficient representation  $\mathbf{u}$  of a set of basis vectors  $\mathbf{Q}$ , i.e.,  $\mathbf{Q}\mathbf{u} = \mathbf{b}$ . As the basis vectors in the transform matrix  $\mathbf{Q}$  are usually overcomplete, we introduce additional sparsity criterion to address the degeneracy problem caused by overcompleteness. Depending on how to force the algorithm to provide a satisfactory representation of  $\mathbf{b}$ , sparse coding can be considered as a constrained or regularized problem. Note that in both cases, usually we set the objective function  $\ell$  in Eq. (3) to be the MSE loss.

**Constrained Sparse Coding.** The constrained sparse coding form is based on linear equality constraints  $\mathbf{Q}\mathbf{u} = \mathbf{b}$ , corresponding to the constraint term in Eq. (1) as a guarantee of reconfigurability. As the reconstruction is usually imperfect, Since the transform matrix  $\mathbf{Q}$  is usually generated from clear data, noise in the given data  $\mathbf{b}$  cannot be perfectly restored, so the noise estimation term  $\mathbf{u}_n$  is added as a complement to adhere to the task information, i.e.,  $\mathbf{Q}\mathbf{u} + \mathbf{u}_n = \mathbf{b}$ . Note that here we need to additionally estimate the noise term  $\mathbf{u}_n$ , and in other cases if the noise is a constant vector, we just denote it to be  $\mathbf{n}$ . As an overcomplete task, the  $\ell_1$  paradigm is usually used as a sparsity penalty which forces our representation of  $\mathbf{u}$  and  $\mathbf{u}_n$  to be sparse. We model the constrained sparse coding problem as the following

$$\min_{\mathbf{u}, \mathbf{u}_n} \kappa \|\mathbf{u}\|_1 + \|\mathbf{u}_n\|_1 \quad \text{s.t.} \quad \mathbf{Q}\mathbf{u} + \mathbf{u}_n = \mathbf{b} \quad (8)$$

where  $\kappa$  is a scaling constant to determine the relative importance of the two norms. In order to solve the constrained optimization problem while satisfying the assumptions of HODL, we use the ALM method to determine  $\mathcal{D}_{\text{ALM}}$  as  $\mathcal{D}_{\text{num}}$  as shown in Table 2. It can be proved that corresponding  $\mathcal{D}_{\text{ALM}}$  for Eq. (8) satisfies Assumption 3.2 under mild conditions. Please refer to [3, Appendix B] for details.

**Regularized Sparse Coding.** Another common type of task prior is to add regularization terms as the learnable module in Eq. (1) to the objective function. The regularized sparse coding form is based on reconstruction term  $\|\mathbf{Q}\mathbf{u} - \mathbf{b}\|_2$  as a guarantee of reconfigurability. As an overcomplete task regularization, it also uses  $\ell_1$  paradigm as a sparsity penalty to force the representation of  $\mathbf{u}$  to be sparse. We define the objective function for regularized sparse coding as

$$\min_{\mathbf{u}} \|\mathbf{Q}\mathbf{u} - \mathbf{b}\|_2 + \kappa \|\mathbf{u}\|_1 \quad (9)$$

where  $\kappa$  is a scaling constant to determine the relative importance between reconstruction term and regularization term. In order to solve the regularized optimization problem while satisfying the assumptions of HODL, we use the PG method to determine  $\mathcal{D}_{\text{PG}}$  as  $\mathcal{D}_{\text{num}}$  as shown in Table 2. In [3, Appendix B], it is proved that corresponding  $\mathcal{D}_{\text{PG}}$  satisfies Assumption 3.2 under mild conditions.

**Composition of  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$ .** In the above discussion we use a fully connected layer network with ReLU activation and spectral normalization as  $\mathcal{D}_{\text{net}}$ . When compositing  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$  for better performance, we use a non-expansive

TABLE 2: Summary of operator  $\mathcal{D}_{\text{num}}$ ,  $\mathcal{D}_{\text{net}}$ , and applications for various models. Here NE-net denotes Non-Expansive networks (1-Lipschitz continuous, with respect to  $\|\cdot\|_{\mathbf{G}_\omega}$ ), and GD is short of gradient descent. Note that here N/A means that  $\mathcal{D}_{\text{num}}$  or  $\mathcal{D}_{\text{net}}$  is not employed for the corresponding tasks following the common settings.

Model	$\mathcal{D}_{\text{num}}$	$\mathcal{D}_{\text{net}}$	Applications
Constrained Problems	ALM : $\begin{cases} \mathbf{u}^{k+1} = \underset{\mathbf{u}}{\operatorname{argmin}} \left\{ f(\mathbf{u}) + \langle \lambda^k, \mathcal{A}(\mathbf{u})\mathbf{u} - \mathbf{y}(\omega) \rangle + \frac{\beta}{2} \ \mathcal{A}(\mathbf{u})\mathbf{u} - \mathbf{y}(\omega)\ ^2 + \frac{1}{2} \ \mathbf{u} - \mathbf{u}^k\ _{\mathbf{G}_\omega}^2 \right\} \\ \lambda^{k+1} = \lambda^k + \beta(\mathcal{A}(\mathbf{u})\mathbf{u}^{k+1} - \mathbf{y}(\omega)) \end{cases}$	NE-net	Sparse Coding Rain Streak Removal
Regularized Problems	PG : $\mathbf{u}^{k+1} = \underset{\mathbf{u}}{\operatorname{argmin}} \left\{ f(\mathbf{u}^k) + \langle \nabla_{\mathbf{u}} f(\mathbf{u}^k), \mathbf{u} - \mathbf{u}^k \rangle + g(\mathbf{u}, \omega) + \frac{1}{2\gamma} \ \mathbf{u} - \mathbf{u}^k\ _{\mathbf{G}_\omega}^2 \right\}$		Sparse Coding Image Deconvolution Low-light Enhancement
Hierarchical Models	GD : $\mathbf{u}^{k+1} = \mathbf{u}^k - \nabla_{\mathbf{u}} f(\mathbf{u}^k)$	N/A	Hyper-parameter Optimization Few-shot Learning
	N/A	NE-net	Adversarial Learning

$\mathcal{D}_{\text{net}}$  as shown in Table 2 and composite them to satisfy the assumptions of HODL solution strategy.

The convergence guarantee will hold when compositing  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$ , because if  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$  satisfy Assumption 3.2(or 3.4) with the same  $\mathbf{G}_\omega$ , then  $\mathcal{D}_{\text{num}} \circ \mathcal{D}_{\text{net}}$  also satisfies these assumptions. To be specific, the non-expansive (or contractive) property of  $\mathcal{D}_{\text{num}} \circ \mathcal{D}_{\text{net}}$  with  $\mathbf{G}_\omega$  can be easily verified from the definition. As for the closeness of  $\mathcal{D}_{\text{num}}(\cdot, \omega) \circ \mathcal{D}_{\text{net}}(\cdot, \omega)$  for a fixed  $\omega \in \Omega$ , we consider the sequence  $\{(\mathbf{u}^k, \mathbf{v}^k)\} \in \operatorname{gph}(\mathcal{D}_{\text{num}}(\cdot, \omega) \circ \mathcal{D}_{\text{net}}(\cdot, \omega))$  satisfying  $(\mathbf{u}^k, \mathbf{v}^k) \rightarrow (\bar{\mathbf{u}}, \bar{\mathbf{v}})$ . From the boundedness of  $\{\mathbf{u}^k\}$  and the non-expansive (or contractive) property of  $\mathcal{D}_{\text{num}} \circ \mathcal{D}_{\text{net}}$  with  $\mathbf{G}_\omega \succ 0$ , it can be obtained that  $\mathcal{D}_{\text{net}}(\mathbf{u}^k, \omega)$  is bounded, so there exists a subsequence  $\{(\mathbf{u}^i, \mathbf{v}^i)\} \subseteq \{(\mathbf{u}^k, \mathbf{v}^k)\}$  such that  $\mathcal{D}_{\text{net}}(\mathbf{u}^i, \omega) \rightarrow \bar{\omega}$ . Then it follows from the closeness of  $\mathcal{D}_{\text{net}}(\cdot, \omega)$  and  $\mathcal{D}_{\text{num}}(\cdot, \omega)$  that  $(\bar{\mathbf{u}}, \bar{\omega}) \in \operatorname{gph} \mathcal{D}_{\text{net}}(\cdot, \omega)$  and  $(\bar{\omega}, \bar{\mathbf{v}}) \in \operatorname{gph} \mathcal{D}_{\text{num}}(\cdot, \omega)$ . Hence,  $(\bar{\mathbf{u}}, \bar{\mathbf{v}}) \in \operatorname{gph}(\mathcal{D}_{\text{num}}(\cdot, \omega) \circ \mathcal{D}_{\text{net}}(\cdot, \omega))$ . Note that given any non-expansive  $\mathcal{D}_{\text{net}}$  (which can be achieved by spectral normalization) and positive-definite matrix  $\mathbf{G}_\omega$ , by setting  $\mathcal{D}_{\text{net}^*} = \mathbf{G}_\omega^{-1/2} \mathcal{D}_{\text{net}} \mathbf{G}_\omega^{1/2}$ , then  $\mathcal{D}_{\text{net}^*}$  satisfies Assumption 3.2(or 3.4) with  $\mathbf{G}_\omega$ .

### 4.3 Applications for Vision Tasks

In this subsection, we illustrate the applications of ODL in vision tasks, describe the shortcomings of existing ODL methods, and demonstrate how to apply HODL in vision tasks. In these applications, we use  $\mathcal{D}_{\text{ALM}}$  and  $\mathcal{D}_{\text{PG}}$  as  $\mathcal{D}_{\text{num}}$  for constrained and regularized problems, respectively, consistent with the discussion for sparse coding. In addition, we set the objective function  $\ell$  in Eq. (3) to be the MSE loss.

**Rain Streak Removal.** An application scenario of constrained HODL requires using variable separation to aid in problem solving. As an example, in the rain streak removal task, the sparse solutions of rain line and background are solved separately by adding auxiliary variables [35]. This scenario requires the auxiliary variables and the original variables to be kept equal, and it is suitable to use HODL framework with equality constraints. Specifically, given the input rainy image  $\mathbf{I}_r$ , the goal is to decompose it into a rain-free background  $\mathbf{u}_b$  and a rain streak layer  $\mathbf{u}_r$ , i.e.,  $\mathbf{I}_r = \mathbf{u}_b + \mathbf{u}_r$ , to enhance the visibility. The problem can be reformulated as  $\min_{\mathbf{u}_b, \mathbf{u}_r} \frac{1}{2} \|\mathbf{u}_b + \mathbf{u}_r - \mathbf{I}_r\|_2^2 + \psi_b(\mathbf{u}_b) + \psi_r(\mathbf{u}_r)$ , where  $\psi_b(\mathbf{u}_b)$  and  $\psi_r(\mathbf{u}_r)$  are set to be  $\psi_b(\mathbf{u}_b) = \kappa_b \|\mathbf{u}_b\|_1$  and  $\psi_r(\mathbf{u}_r) = \kappa_r \|\nabla \mathbf{u}_r\|_1$ , representing the priors on the background layer and rain streak layer respectively. Then we introduce auxiliary variables  $\mathbf{v}_b$  and  $\mathbf{v}_r$ , and transfer the

problem to be  $\min_{\mathbf{u}_b, \mathbf{u}_r, \mathbf{v}_b, \mathbf{v}_r} \frac{1}{2} \|\mathbf{u}_b + \mathbf{u}_r - \mathbf{b}\|_2^2 + \kappa_b \|\mathbf{v}_b\|_1 + \kappa_r \|\mathbf{v}_r\|_1$ , s.t.,  $\mathbf{v}_b = \mathbf{u}_b$ ,  $\mathbf{v}_r = \nabla \mathbf{u}_r$ , where  $\nabla = [\nabla_h; \nabla_v]$  denotes the gradient in horizontal and vertical directions. Existing ODL methods usually solve  $\mathbf{u}_b, \mathbf{u}_r$  using  $\mathcal{D}_{\text{num}}$  and solve  $\mathbf{v}_b, \mathbf{v}_r$  by a pre-trained  $\mathcal{D}_{\text{net}}$ , usually leading to a gap between the pre-trained task and current task. HODL, in contrast, ensures that  $\mathcal{D}_{\text{net}}$  learns valid rain streak information by using a regularized  $\mathcal{D}_{\text{net}}$  trained on current task jointly with  $\mathcal{D}_{\text{num}}$ .

**Image Deconvolution.** As an application of regularized HODL, image deconvolution does not strive for perfect image restoration, but pursues a balance between restoration and deconvolution effects whenever possible [12]. Specifically, the input image can be expressed as  $\mathbf{b} = \mathbf{Q} * \mathbf{u} + \mathbf{n}$ , where  $\mathbf{Q}, \mathbf{u}$ , and  $\mathbf{n}$  respectively denote the blur kernel, latent clean image, and additional noise, and  $*$  denotes the two-dimensional convolution operator. Here the regularization is implemented based on Maximum A Posteriori (MAP) estimation. Then the problem is transferred to  $\min_{\mathbf{u} \in U} \|\mathbf{Q} * \mathbf{u} - \mathbf{b}\|_2^2 + g(\mathbf{u})$ , where  $g(\mathbf{u})$  is the prior function of the image. We set  $g(\mathbf{u})$  to be  $\kappa \|\mathbf{W}\mathbf{u}\|_1$ , where  $\mathbf{W}$  is the wavelet transform matrix, considering that there is usually a sparse image after the wavelet transform. In this task, existing ODL approaches typically have two ideas. One uses  $\mathcal{D}_{\text{num}}$  for task fidelity term  $\|\mathbf{Q} * \mathbf{u} - \mathbf{b}\|_2^2$  and pre-trained  $\mathcal{D}_{\text{net}}$  for regularization term  $g(\mathbf{u})$  to guarantee clarity. Similar to the previous task, this makes the pre-trained  $\mathcal{D}_{\text{net}}$  not well adapted to the current task details such as convolution kernels and object edges. The other is to train  $\mathcal{D}_{\text{net}}$  in the current task, but ignore  $\mathcal{D}_{\text{num}}$  during training after having built  $\mathcal{D}_{\text{net}}$  from  $\mathcal{D}_{\text{num}}$ . HODL, on the other hand, ensures  $\mathcal{D}_{\text{num}}$  to control over the iteration and enables  $\mathcal{D}_{\text{net}}$  to adapt to the current task through joint training.

**Low-light Enhancement.** As another application of regularized HODL, low-light enhancement usually employs a complex network to estimate illumination in order for a higher image quality. Hence, compared with linear equality constraint terms, it is more appropriate to use regularization terms as a priori. Specifically, we follow the simple Retinex rule  $\mathbf{y} = \mathbf{x} \otimes \mathbf{u}$ , where  $\mathbf{y}$  is the captured underexposed observation which is a given low-light image,  $\mathbf{x}$  is the desired recovery,  $\mathbf{u}$  is the illumination to be determined for enhancement, and the operator  $\otimes$  denotes element-wise multiplication. To accurately estimate  $\mathbf{u}$ , inspired by the work in [36], we estimate  $\mathbf{u}$  by  $\min_{\mathbf{u}} \|\mathbf{u} - \phi(\mathbf{y})\|_2^2 + \psi(\mathbf{u})$ , where  $\phi$  is a given estimated illumination mapping, and  $\psi$  is a regularization function estimated implicitly from a CNN.

In this task, existing ODL methods usually construct the network for solving task term  $\|\mathbf{u} - \phi(\mathbf{y})\|_2^2$  and regularization term  $\psi(\mathbf{u})$  from an optimization problem, but along with the training procedure, the network structure will be away from the original optimization structure. However, HODL is able to retain the optimization structure in training, thus effectively improving the image fidelity.

## 5 EXTENSIONS TO OTHER LEARNING TASKS

In this section we illustrate how to apply the hierarchical modeling of HODL to a wide range of learning tasks beyond ODL. Specifically, as a methodology, HODL framework with hierarchical structures is not limited to specific methods and can be used to uncover the hierarchical relationships in multi-task coupled learning tasks as well. Since learning tasks can be considered as optimization problems based on loss functions and specific optimizers, HODL, which is dedicated to modeling hierarchical relationships between optimization and learning, can also accommodate hierarchical coupling in multiple learning tasks. For example, by setting the optimization operator  $\mathcal{T}$  in Eq. (3) as the gradient descent operator for optimizing the sub-task loss function, and  $\ell$  in Eq. (3) as the loss function for another sub-task, HODL can be easily migrated to any learning application with multiple sub-tasks. Actually, bilevel optimization can be regarded as a special case of HODL framework, if we restrict  $\mathcal{T}$  to be the operators for solving optimization problems. More specifically, for adversarial learning,  $\ell$  is used to characterize the antagonistic relationship between the generator and discriminator; for hyper-parameter optimization and few-shot learning,  $\ell$  is the cross entropy loss function on the validation set. Please also refer to [37] for more detailed expression of  $\ell$  and  $\mathcal{T}$ . Therefore, HODL can be widely applied in adversarial learning [38], [39], hyper-parameter optimization [37], [40], few-shot learning [32], and so on, as shown in Table 2.

**Adversarial Learning.** As the best-known application of adversarial learning, Generative Adversarial Networks (GAN) has received much attention in recent years, which adversarially trains generators to solve real-world tasks by means of additional discriminators. In GAN, the generator depends on the discrimination from the discriminator to learn the features, while the discriminator depends on the output of generator to learn the classification. Therefore, by taking the update of discriminator as the operator  $\mathcal{T}$  in Eq. (3) and the learning process of generator as  $\ell$  in Eq. (3), HODL can effectively model the coupling relationship between the two sub-tasks of GAN.

**Hyper-parameter Optimization.** The increasing complexity of machine learning algorithms has driven plenty of research in the field of hyper-parameter optimization. In machine learning, hyper-parameter optimization aims at choosing a set of optimal hyper-parameters for learning algorithms. Hyper-parameters are a class of parameters whose values are used to control the learning process. Therefore, by taking the learning process as the operator  $\mathcal{T}$  in Eq. (3) and the objective function to choose optimal hyper-parameters as  $\ell$  in Eq. (3), our HODL approach is equally effective when dealing with hyper-parameter optimization.

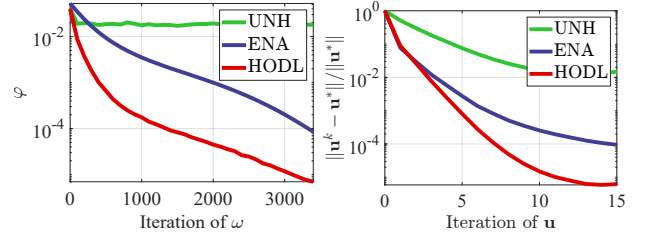


Fig. 1: The convergence behavior of  $\omega$  and  $\mathbf{u}$  by UNH, ENA, and HODL for regularized sparse coding. It can be seen that for regularized problems using PG, our HODL has better convergence results.

**Few-shot Learning.** Few-shot learning ( $N$ -way  $M$ -shot) is a multi-task  $N$ -way classification which aims to learn the feature extraction structure with generalization ability, so that each new task can be solved only through  $M$ -training samples. This task has nested hierarchies, which respectively classify  $M$  samples and learn a feature structure that can be used for new tasks. Therefore, by taking the classification optimization process as the operator  $\mathcal{T}$  in Eq. (3) and the learning process of feature structure as function  $\ell$  in Eq. (3), our HODL approach can also be applied.

Besides, in some applications, the operator  $\mathcal{T}$  corresponds to optimizing an implicit energy function that is solved indirectly through a neural network. In this case, by applying spectral normalization to the network, we can still obtain a non-expansive mapping. We verify the necessity of the non-expansive property of neural network in Section 6.1.

## 6 EXPERIMENTAL RESULTS

In this section, we first verify the theoretical properties of HODL on synthetic experiments in the sparse coding task. We subsequently apply HODL to visual experiments containing rain streak removal, image deconvolution, and low-light enhancement. Finally, we extend HODL to other applications with hierarchies, including adversarial learning, hyper-parameter optimization, and few-shot learning. We conduct our experiments mainly on a PC with Intel Core i9-10900KF CPU (3.70GHz), 128GB RAM and two NVIDIA GeForce RTX 3090 24GB GPUs. All experiments are implemented on synthetic datasets, and the Adam optimizer is adopted to update variable  $\omega$ . Note that other acceleration techniques are also applicable under the HODL framework.

### 6.1 Model Evaluation

This part first verifies that HODL improves the overall performance compared with existing ODL methods. More specifically, we analyze the performance on convergence by HODL in terms of learning variables and optimization variables for the learning process and optimization process, respectively. After that, we investigate some factors that may affect the performance of HODL. To illustrate the generality of HODL, we verify the performance on constrained and regularized sparse coding problems.

For regularized problems, we use the regularized sparse coding model introduced in Section 4.2. We set  $m = 500$ ,  $n = 250$  ( $\mathbf{Q}$  in Eq. (9) is a  $m \times n$  matrix), and the training and testing samples are 10000 and 1000, respectively.



TABLE 3: PSNR and SSIM results for constrained sparse coding on Set14. Best and second best results are marked in red and blue respectively.

Methods	Layers	PSNR	SSIM
UNH	5	10.47 $\pm$ 2.36	0.41 $\pm$ 0.14
	25	11.31 $\pm$ 2.29	0.41 $\pm$ 0.15
ENA	5	15.59 $\pm$ 0.81	0.52 $\pm$ 0.13
	25	15.64 $\pm$ 0.87	0.52 $\pm$ 0.13
HODL	5	<b>18.82<math>\pm</math>1.59</b>	<b>0.63<math>\pm</math>0.16</b>
	25	<b>18.98<math>\pm</math>2.53</b>	<b>0.65<math>\pm</math>0.15</b>

The elements of matrix  $\mathbf{Q}$  are sampled from the standard Gaussian distribution, and the column vector of matrix  $\mathbf{Q}$  is standardized to have the unit  $\ell_2$  norm. The sparse vector  $\mathbf{u}$  is sampled from the standard Gaussian distribution, and the distribution of non-zero elements follows the Bernoulli distribution with probability 0.1. The intensity of noise  $\mathbf{n}$  is 0.01 times the standard Gaussian distribution, and all data are generated by the model  $\mathbf{b} = \mathbf{Q}\mathbf{u} + \mathbf{n}$ . To be fair for the comparisons,  $\mathbf{Q}$  and  $\mathbf{b}$  are fixed in the experiment. We use the MSE loss as the supervised loss. Note that here  $\mathcal{D}_{\text{num}}$  is  $\mathcal{D}_{\text{pg}}$ , and  $\mathcal{D}_{\text{net}}$  is a fully connect network with ReLU activation. For comparison, UNH stands for the method that only learns the step size, while ENA stands for the method that learns  $\mathcal{D}_{\text{net}}$ . To show the performance of HODL for regularized problems, we compare the convergence of different methods in the optimization process for  $\mathbf{u}$  and learning process for  $\omega$  in Figure 1. It can be seen that, HODL performs better in the convergence of optimization and learning than other methods.

For constrained problems, we follow the setting in [12] to use the classic Set14 dataset as experimental data, in which the salt-and-pepper noise is added to 10% pixels of each image. The rectangle of each image is divided into non-overlapping patches of size  $16 \times 16$ . We use the patch dictionary method to learn a  $256 \times 512$  dictionary  $\mathbf{Q}$ . We set batch size = 128, training set size = 10000, and random seed = 1126. The testing set size depends on the size of each image. Because we conduct unsupervised single image training, we do not use the MSE loss between the clear picture and the generated picture, but instead use the same unsupervised loss as in [13].

To show the performance of HODL for constrained sparse coding, we present the PSNR and SSIM results in Table 3. It can be seen that the performance of our HODL on both PSNR and SSIM is superior than UNH and ENA. This is because UNH can only train few learning variables (such as the step size) to maintain convergence, and the neglect of the original optimization structure during training by ENA brings about a distance from the real fixed point model. In contrary, thanks to the hybrid strategy to incorporate optimization and learning processes, HODL allows more learning variables to improve the performance. Considering the consistent performance of constrained HODL and regularized HODL, for simplicity, we base our subsequent analysis on the constrained HODL.

To illustrate in detail how HODL improves the performance of ODL, we next analyze the convergence of learning variables  $\omega$  and optimization variables  $\mathbf{u}$ , respectively. In Figure 2, we first analyze the convergence behavior of

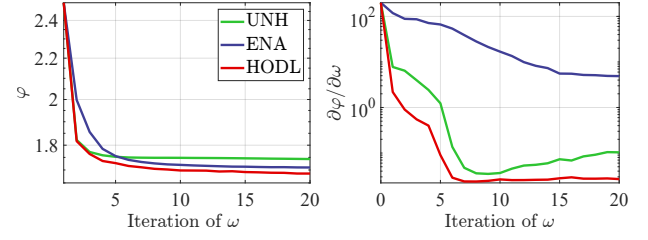


Fig. 2: The convergence curves of  $\varphi$  and  $\partial\varphi/\partial\omega$  with respect to  $\omega$  for constrained sparse coding. UNH does not add learnable knowledge to optimization and ENA ignores the optimization structure during training. It can be seen that our method achieves the optimal convergence of loss function with a stationary gradient curve.

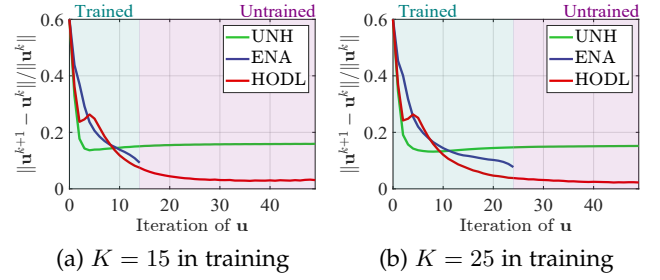


Fig. 3: The convergence curves of  $\|\mathbf{u}^{k+1} - \mathbf{u}^k\|/\|\mathbf{u}^k\|$  with respect to  $\mathbf{u}$  after (a)  $K = 15$  and (b)  $K = 25$  as iterations of  $\mathbf{u}$  in training, where  $k$  is the number of iterations of  $\mathbf{u}$  for optimization in testing. The green background indicates when the training iteration of  $\mathbf{u}$  is less than testing iteration, while the pink background represents the testing iteration is beyond training iteration. It can be seen that our method can successfully learn the non-expansive mapping and converge better after different iterations in training.

learning variables  $\omega$  in the objective function of learning process  $\varphi_K(\omega) = \ell(\mathbf{u}^K(\omega), \omega)$  defined in Eq. (7) with a fixed  $K$ . ENA and UNH perform poorly in the convergence of learning objective function, while HODL is able to effectively obtain better convergence.

Next, we verify the convergence of optimization variables  $\mathbf{u}$ . On one hand, from the green background part of Figure 3, it can be seen that HODL outperforms other methods in convergence stability and convergence speed. UNH converges fast at first, but it cannot further improve the convergence performance. ENA has slow convergence speed because its neglect of optimization structure during training. Conversely, HODL can effectively reduce the required number of iterations in training to control the expected error, which increases the computational efficiency. On the other hand, in practical applications, limited by the high computational burden on training time, one tends to train in a smaller number of optimization iterations and subsequently expects to obtain higher performance in testing. This requires ODL methods to be able to learn a stable non-expansive mapping. Therefore, we additionally observe the convergence curves of the optimization variables  $\mathbf{u}$  in this case to further verify the stability and non-expansive property of the trained optimization iterative module. In Figure 3, we also show the convergence curve when the

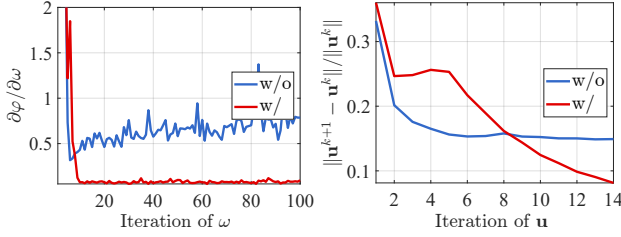


Fig. 4: Convergence curves of  $\partial\varphi/\partial\omega$  with respect to  $\omega$  and  $\|u^{k+1} - u^k\|/\|u^k\|$  with respect to  $u$ , with or without the non-expansive property of operator  $\mathcal{D}$  in HODL. The necessity of non-expansive property of  $\mathcal{D}$  for HODL can be observed.

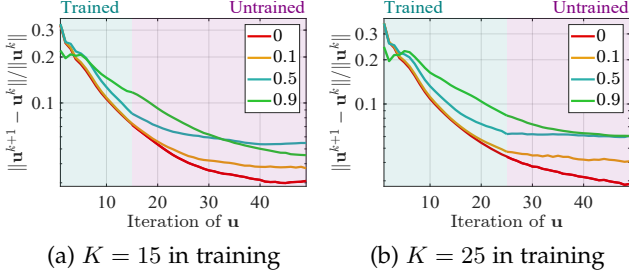


Fig. 5: Convergence curves of  $\|u^{k+1} - u^k\|/\|u^k\|$  with different  $\mu$  in Eq. (4) (aHODL) in Algorithm 1. Note that HODL with  $\mu = 0$  is equivalent to sHODL. In the case of complex networks, sHODL is more likely to achieve satisfying performance early in the iteration.

number of optimization iterations of  $u$  in testing is more than those in training (the pink background part). Note that since for ENA the number of iterations of  $u$  is fixed in training, it cannot be compared in this case. Still, we find that HODL is superior to UNH, and the mapping learned by our HODL can indeed continue to converge in the testing iterations beyond training steps, implying that we have effectively learned a non-expansive mapping with convergence.

In addition, we investigate some factors that may affect the performance of HODL, including the necessity of non-expansive property, and the influence of parameter  $\mu$  on convergence. In Figure 4, we verify the effect of non-expansive property of  $\mathcal{D}$  on the convergence. It can be seen that the non-expansive property reduces the gradient of the learning objective  $\varphi$  by an order of magnitude, increases the convergence stability, and also provides a better convergence of the optimization iteration. These verify the importance of the non-expansive property on the convergence. In Figure 5, we show the impact of different values of parameter  $\mu$  in Eq. 4 (aHODL) in Algorithm 1 on the convergence of optimization process. We can find that with the iteration of  $u$ , the best performance is obtained for smaller  $\mu$  and even for the smallest 0 (i.e., sHODL in Eq. 5). Comparing (a) and (b) in Figure 5, it can be seen that when the network is relatively more fully trained, the advantage of choosing an appropriate  $\mu$  is more obvious. Figure 6 further compares the computational efficiency of sHODL and aHODL, which shows that sHODL is less computationally intensive than aHODL. Therefore, considering the computational complexity burden in practical applications, we focus on sHODL from now on, including in the subsequent experiments of

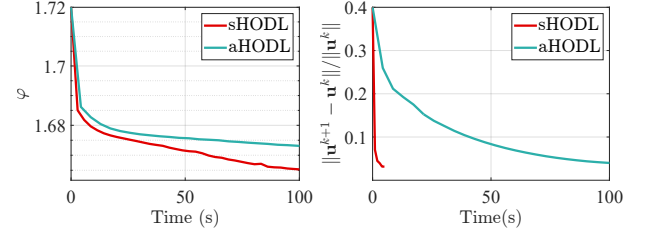


Fig. 6: Computational efficiency of sHODL and aHODL.

TABLE 4: PSNR results of constrained sparse coding using ALM and ADMM. ADMM performs better than ALM for UNH and ENA, but our approach achieves the best performance only using ALM. Best and second best results are marked in red and blue respectively.

Methods	UNH	ENA	HODL
ADMM	11.27±2.71	<b>15.58±0.89</b>	N/A
ALM	7.32±4.65	13.78±5.23	<b>18.64±0.74</b>

vision tasks and extended applications.

Finally, it should be noted that for constrained problems, existing methods typically use ALM or ADMM, while HODL uses ALM as  $\mathcal{D}_{\text{num}}$ . For the fairness of comparisons, we examine the performance of UNH and ENA using ALM and ADMM, and HODL using ALM. As can be seen in Table 4, the performance using ALM is weaker than ADMM in both existing UNA and ENA methods, while our HODL only using ALM is able to outperform other methods, further demonstrating the effectiveness of HODL. Actually, aforementioned experiments of UNH and ENA for comparison are conducted using ADMM as the base method.

## 6.2 Vision Tasks

This subsection provides experimental results in vision tasks including rain streak removal, image deconvolution, and low-light enhancement.

**Rain Streak Removal.** In the rain streak removal task, we use datasets Rain100L and Rain100H [41]. As a constrained problem,  $\mathcal{D}_{\text{num}}$  is set to be  $\mathcal{D}_{\text{ALM}}$ . For the network architecture  $\mathcal{D}_{\text{net}}$ , we adopt a 2-layer convolutional network with  $u_r$  and  $b$  as the network input to estimate  $u_b$ , and a 3-layer

TABLE 5: Averaged PSNR and SSIM results for the single image rain removal task on two widely used synthesized datasets, Rain100L and Rain100H [41]. Best and second best results are marked in red and blue respectively.

Datasets Metrics	Rain 100L		Rain 100H	
	PSNR	SSIM	PSNR	SSIM
DSC (ENA)	27.34	0.849	13.77	0.319
GMM (ENA)	29.05	0.871	15.23	0.449
JCAS	28.54	0.852	14.62	0.451
Clear	30.24	0.934	15.33	0.742
DDN	32.38	0.925	22.85	0.725
RESCAN	38.52	0.981	29.62	0.872
PReNet (ENA)	37.45	0.979	30.11	<b>0.905</b>
SPANet	35.33	0.969	25.11	0.833
JORDER_E	38.59	<b>0.983</b>	30.50	0.896
SIRR	32.37	0.925	22.47	0.716
MPRNet	36.40	0.965	30.41	0.890
RCDNet (ENA)	<b>40.00</b>	<b>0.986</b>	<b>31.28</b>	<b>0.903</b>
HODL	<b>40.07</b>	<b>0.986</b>	<b>30.96</b>	<b>0.905</b>



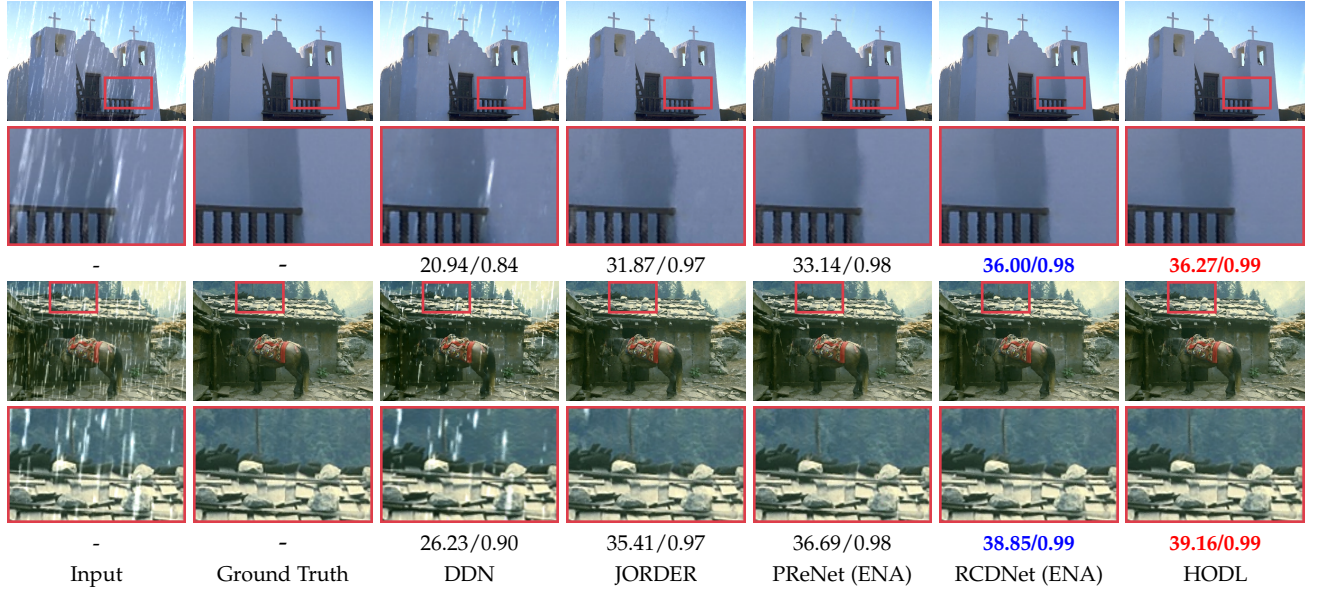


Fig. 7: Visual results of the rain streak removal task on two samples from Rain100L, compared with DDN, JORDER, PReNet and RCDNet. The hierarchical structure of HODL reduces the distortion and blur introduced by removing rain lines. Two metrics (PSNR / SSIM) are listed below each image to quantify the quality of generated images. Best and second best results are marked in red and blue respectively.

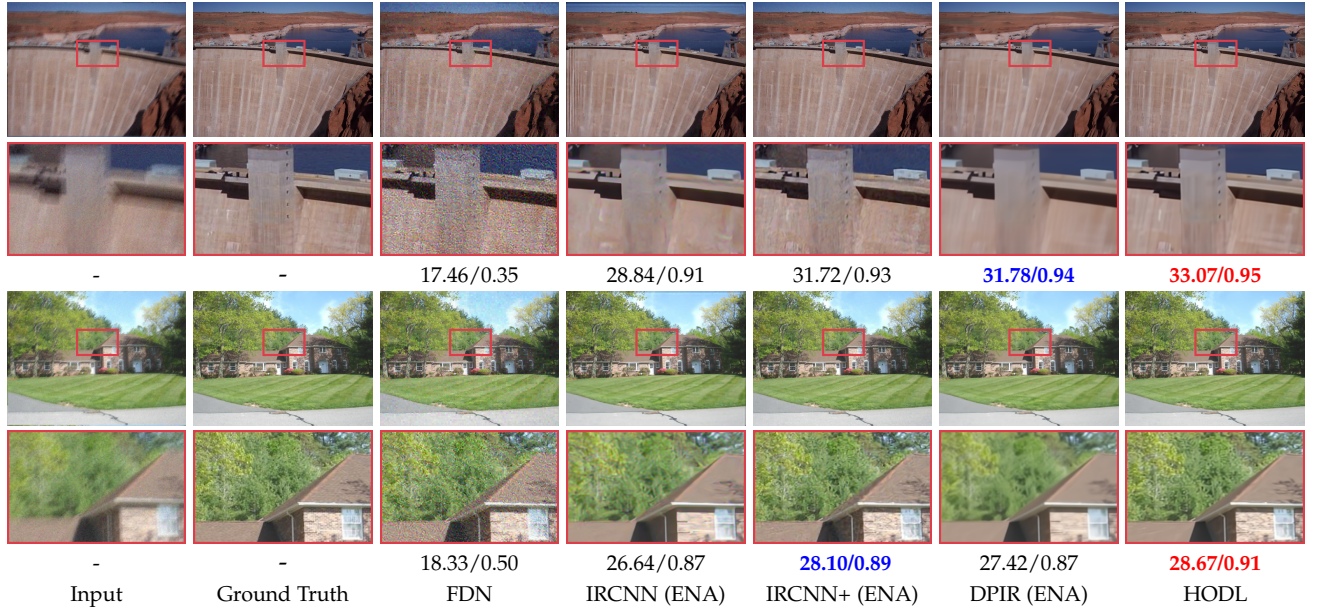


Fig. 8: Visual results of the image deconvolution task on two samples, compared with FDN, IRCNN, IRCNN+, and DPIR. The hierarchical modeling of HODL improves the clarity of details and maintains the high level of color restoration. Two metrics (PSNR / SSIM) are listed below each image to quantify the quality of generated images. Best and second best results are marked in red and blue respectively.

convolutional network with  $\mathbf{u}_b$ ,  $\mathbf{u}_r$ , and  $\mathbf{b}$  as the input to estimate  $\mathbf{u}_r$ . In the network to estimate  $\mathbf{u}_r$ , some prior information of  $\mathbf{u}_r$  is employed as input just like in [42]. In practice, we decide proper  $\Omega$  such that for all  $\omega \in \Omega$  it holds that  $\mathbf{G}_\omega \succ 0$ , and  $\mathbf{G}_\omega$  can be inverted fast by Fourier transform. Here we use MSE as the loss function and use Adam optimizer with step size 0.001, and set batchsize = 64.

We report the quantitative comparison of HODL in Table 5 with a series of state-of-the-art methods. It can

be seen that on both benchmark datasets HODL achieves higher PSNR and SSIM. Note that HODL has a competitive performance compared with RCDNet, and it possesses superior theoretical property as well. In Figure 7, we visually present the performance of rain streak removal task on two images from Rain100L [41], compared with DDN [43], JORDER [41], PReNet [44] and RCDNet [42]. From both rows, one can observe that our HODL preserves the original



TABLE 6: PSNR (dB) results compared with state-of-the-art methods for the image deconvolution task with noise levels  $\sigma = 1\%$  and  $3\%$ . Best and second best results are marked in red and blue respectively.

Noise level Image	$\sigma = 1\%$			$\sigma = 3\%$		
	Butterfly	Leaves	Starfish	Butterfly	Leaves	Starfish
EPLL	20.55	19.22	24.84	18.64	17.54	22.47
FDN	27.40	26.51	27.48	24.27	23.53	24.71
IRCNN (ENA)	32.74	33.22	33.53	28.53	28.45	28.42
IRCNN+ (ENA)	32.48	33.59	32.18	28.40	28.14	28.20
DPIR (ENA)	<b>34.18</b>	<b>35.12</b>	<b>33.91</b>	<b>29.45</b>	<b>30.27</b>	<b>29.46</b>
HODL	<b>33.67</b>	<b>35.39</b>	<b>33.98</b>	<b>29.46</b>	<b>30.69</b>	<b>29.64</b>

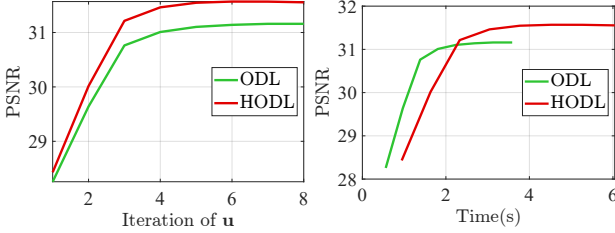


Fig. 9: PSNR results of ODL (DPIR which can be classified as ENA) and HODL with iteration of  $\mathbf{u}$  and inference time for the image deconvolution task.

counter line of wall and roof in the background and performs the best on PSNR and SSIM, while other methods produce some unsatisfactory distortion, blur some textures, or even leave noticeable rain streaks.

**Image Deconvolution.** In the image deconvolution task, similar to [16], we use a large dataset containing 400 images from Berkeley Segmentation Dataset, 4744 images from Waterloo Exploration Database, 900 images from DIV2K Dataset, and 2750 images from Flickr2K Dataset. As for the network architectures  $\mathcal{D}_{\text{net}}$ , we use DRUNet containing four scales, each of which has an identity skip connection between  $2 \times 2$  strided convolution downscaling and  $2 \times 2$  transposed convolution upscaling operators. From the first scale to the fourth scale, the numbers of channels in each layer are respectively 64, 128, 256, and 512. We employ four successive residual blocks in the downscaling and upscaling of each scale. For the numerical update operator  $\mathcal{D}_{\text{num}}$ , by introducing the auxiliary variable  $\mathbf{z} = \mathbf{W}\mathbf{u}$ , we transform the objective function to be  $\|\mathbf{Q}\mathbf{W}^{-1}\mathbf{z} - \mathbf{b}\|_2^2$  with a regularization term  $\|\mathbf{z}\|_1$ . Here we use MSE as the loss function for  $\omega$ , use downsample mode as strideconv, upsample mode as convtranspose, and Adam optimizer with step size 0.001, and set batchsize = 1.

For the practical application in image deconvolution, we verify the performance of HODL on three classical testing images in Table 6, and compare our method with representative methods. For traditional methods, we compare with numerically designed method EPLL [45] and learning-based method FDN [46]. For ODL methods, we compare with IRCNN, IRCNN+, and DPIR [47], [16]. By applying a meta-optimization perspective on handcrafted network  $\mathcal{D}_{\text{net}}$  and numerical schemes  $\mathcal{D}_{\text{PG}}$  as a regularized problem, HODL performs best in the last five columns and achieves top two in the first, in three testing images of different noise levels.

Note that here we choose DRUNet in DPIR [16] as  $\mathcal{D}_{\text{net}}$  for HODL, and the overall preferable results of HODL than directly using DPIR demonstrate the effect of compositing of  $\mathcal{D}_{\text{num}}$  and  $\mathcal{D}_{\text{net}}$  and the ability of HODL to improve the performance based on previous methods. In addition, we show the visual results in Figure 8. It can be seen that our method is superior to other methods in color restoration, detail retention and quantitative metrics. Figure 9 further compares the computational efficiency of ODL (DPIR) and HODL. It can be seen that HODL can reduce the number of iterations without performance degradation.

**Low-light Enhancement.** To further verify the effectiveness of our method on low-level vision tasks, we conduct experiments in the low-light enhancement task. Specifically, we perform experiments on two prominent MIT and LOL datasets, and adopt PSNR, SSIM and LPIPS as our evaluated metrics. Here we use MSE as the loss function for  $\omega$ , set batchsize = 2, and use SGD optimizer with step size 0.015. As for  $\mathcal{D}_{\text{net}}$ , by adopting the continuous relaxation technique used in differentiable Neural Architecture Search (NAS) literature, we search the network structure in the search space which includes  $1 \times 1$  and  $3 \times 3$  Convolution,  $1 \times 1$  and  $3 \times 3$  Residual Convolution,  $3 \times 3$  Dilation Convolution with dilation rate of 2,  $3 \times 3$  Residual Dilation Convolution with dilation rate of 2, and Skip Connection [36]. Then we add spectral normalization to ensure the non-expansive property. For a complete evaluation, we compare HODL with MBLEN [48], GLADNet [49], RetinexNet [50], KinD [51], ZeroDCE [52], FIDE [53], EnGAN [54], and DRBN [55]. In the first three rows of Table 7, we evaluate HODL quantitatively on the MIT Adobe 5K dataset as a simple real-world scenario. In the last three rows of Table 7, we also perform a quantitative assessment on the LOL dataset that increases the difficulty of enhancement due to the inclusion of sensible noise as a demonstration on extremely challenging real world scenarios. It can be seen that HODL obtains the best results on both datasets.

### 6.3 Extended Applications

The followings are experimental results on other learning tasks beyond ODL introduced in Section 5 as the extended applications of HODL.

**Adversarial Learning.** In the adversarial learning task, we visualize the two-dimensional mixed Gaussian distribution data to verify the effectiveness of our method. We set batchsize = 32 and use SGD optimizer with step size 0.001. Performance of HODL is investigated compared to current mainstream and well-known GAN architectures which mitigate mode collapse and maintain stable training, including vanilla GAN (VGAN) [56], WGAN [57], ProxGAN [58], LCGAN [59]. Figure 10 visually shows a comparison of results by various methods regarding the number of samples generated. One can find that mainstream GAN methods only capture a part of distributions, getting into severe mode collapse dilemma and failing to achieve satisfactory performance, while our HODL generates all modes and is significantly better than other methods.

**Hyper-parameter Optimization.** In this experiment, we consider a widely used hyper-parameter optimization example, i.e., data hyper-cleaning, to evaluate the HODL.

TABLE 7: Quantitative results (PSNR, SSIM and LPIPS) on the MIT and LOL datasets for low-light enhancement. Best and second best results are marked in red and blue respectively.

Datasets	Metrics	MBLLEN	GLADNet (UNH)	RetinexNet (ENA)	KinD	ZeroDCE	FIDE	EnGAN	DRBN (ENA)	HODL
MIT	PSNR	15.59	16.73	12.69	<b>17.17</b>	16.46	<b>17.17</b>	15.95	15.01	<b>20.54</b>
	SSIM	0.71	<b>0.76</b>	0.64	0.70	<b>0.76</b>	0.70	0.70	<b>0.77</b>	<b>0.77</b>
	LPIPS	0.31	0.69	0.34	0.32	0.23	<b>0.19</b>	0.29	0.21	<b>0.09</b>
LOL	PSNR	13.93	16.19	13.10	14.62	15.51	<b>16.72</b>	15.32	15.83	<b>20.86</b>
	SSIM	0.49	0.61	0.43	0.64	0.55	0.67	<b>0.70</b>	0.64	<b>0.82</b>
	LPIPS	0.70	<b>0.21</b>	0.86	0.68	0.68	0.72	0.51	0.36	<b>0.08</b>

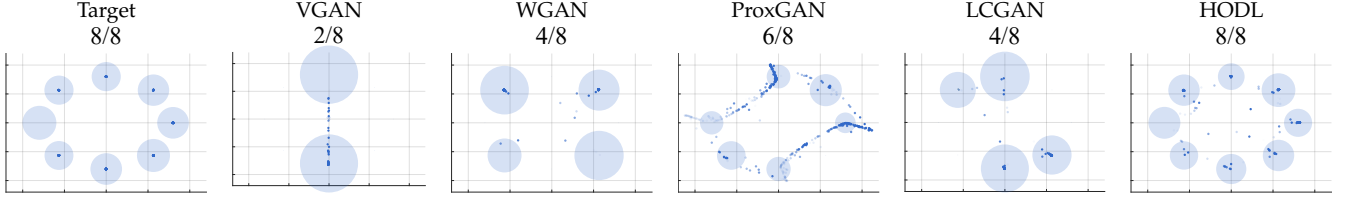


Fig. 10: Comparison among four mainstream GAN methods (i.e., vanilla GAN (VGAN), WGAN, ProxGAN, and LCGAN) and HODL on the synthetic 2D ring mixed of Gaussian distribution data. The gap of generated samples (generated/targeted number of classes) is listed on the top. The shading of dots represents the density of final distribution, with darker dots representing greater density.

TABLE 8: Comparison with existing methods for solving the data hyper-cleaning task on MNIST and FashionMNIST as an example of hyper-parameter optimization. The F1 score denotes the harmonic mean of precision and recall. Best and second best results are marked in red and blue respectively.

Method	MNIST		FashionMNIST	
	Acc.	F1 score	Acc.	F1 score
RHG	87.90	89.36	81.91	87.12
TRHG	88.57	<b>89.77</b>	81.85	86.76
CG	<b>89.19</b>	85.96	<b>83.15</b>	85.13
NS	87.54	89.58	81.37	<b>87.28</b>
HODL	<b>89.75</b>	<b>90.38</b>	<b>82.04</b>	<b>88.24</b>

TABLE 9: The averaged accuracy for few-shot classification on Omniglot and MiniImageNet datasets ( $N$ -way  $M$ -shot with  $M = 1$  and  $N = 5, 20$ ). Best and second best results are marked in red and blue respectively.

Method	Omniglot		MiniImageNet
	5-way	20-way	5-way
RHG	98.60	95.50	<b>48.89</b>
TRHG	98.74	95.82	47.67
CG	<b>98.96</b>	94.46	48.42
NS	98.40	<b>96.06</b>	48.61
HODL	<b>99.04</b>	<b>96.50</b>	<b>49.08</b>

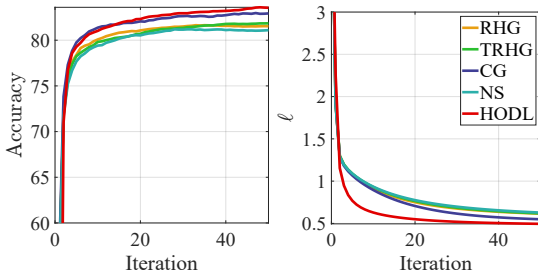


Fig. 11: Comparison of the accuracy and validation loss  $\ell$  for data hyper-cleaning as an example of hyper-parameter optimization with other unrolling algorithms.

Assuming that some labels in our dataset are contaminated, the purpose of data hyper-cleaning is to reduce the impact of incorrect samples by adding hyper-parameters. We follow the settings in [33] and conduct experiments on MNIST and FashionMNIST datasets. We set  $\text{batchsize} = 32$  and use Adam optimizer with step size 0.01. To demonstrate the advantage of our method, we show the accuracy and F1 scores in Table 8, compared with different methods containing Reverse Hyper-Gradient (RHG) [60], Truncated RHG (TRHG) [61], Conjugate

Gradient (CG) [62], and Neumann Series (NS) [63]. Figure 11 also shows the accuracy and validation loss using different methods. It can be seen that our method achieves higher accuracy, higher F1 score, and lower loss.

**Few-shot Learning.** Next, we test the application in few-shot learning under high dimensions on Omniglot and MiniImageNet datasets to verify the computational efficiency of our method. In this experiment, we follow the settings in [33]. We set  $\text{batchsize} = 4$  and use Adam optimizer with step size 0.01. It can be seen in Table 9 that our HODL gives the best performance in different tasks.

## 7 CONCLUSIONS

This paper first proposes the HODL framework to nest the optimization and learning processes in ODL problems, and then presents solution strategies for HODL to jointly solve the optimization variables and learning variables. We prove the joint convergence of optimization variables and learning variables from the perspective of both the approximation quality, and the stationary analysis. Experiments demonstrate our efficiency on sparse coding, real-world applications in image processing (e.g., rain streak removal, image deconvolution, and low-light enhancement), and other learning tasks (e.g.,

adversarial learning, hyper-parameter optimization and few-shot learning). As a flexible and general framework, HODL is also applicable for various networks designed for large-scale problems in real-world applications. Exploring HODL on more large-scale datasets and large neural networks is a future direction.

## ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grants 2020YFB1313503 and 2022YFA1004101, in part by the National Natural Science Foundation of China under Grants U22B2052 and 1222106, in part by Shenzhen Science and Technology Program under Grant RYX20200714114700072, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2022B1515020082, in part by Shandong Province Natural Science Foundation under Grant ZR2023MA020, and in part by Pacific Institute for the Mathematical Sciences (PIMS).

## REFERENCES

- [1] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.
- [2] T. Chen, X. Chen, W. Chen, Z. Wang, H. Heaton, J. Liu, and W. Yin, "Learning to optimize: A primer and a benchmark," *JMLR*, vol. 23, no. 1, pp. 8562–8620, 2022.
- [3] R. Liu, X. Liu, S. Zeng, J. Zhang, and Y. Zhang, "Optimization-derived learning with essential convergence analysis of training and hyper-training," in *ICML*. PMLR, 2022, pp. 13 825–13 856.
- [4] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*. Springer, Cham, 2019, pp. 3–33.
- [5] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 847–855.
- [6] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf, "Learning to deblur," *IEEE TPAMI*, vol. 38, no. 7, pp. 1439–1451, 2015.
- [7] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE TPAMI*, vol. 39, no. 6, pp. 1256–1272, 2016.
- [8] P. Ablin, T. Moreau, M. Massias, and A. Gramfort, "Learning step sizes for unfolded sparse coding," in *NeurIPS*, 2019, pp. 13 100–13 110.
- [9] R. Liu, S. Cheng, L. Ma, X. Fan, and Z. Luo, "Deep proximal unrolling: Algorithmic framework, convergence analysis and applications," *IEEE TIP*, vol. 28, no. 10, pp. 5013–5026, 2019.
- [10] R. Liu, S. Cheng, Y. He, X. Fan, Z. Lin, and Z. Luo, "On the convergence of learning-based iterative methods for nonconvex inverse problems," *IEEE TPAMI*, vol. 42, no. 12, pp. 3027–3039, 2019.
- [11] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *ICML*, 2010, pp. 399–406.
- [12] X. Chen, J. Liu, Z. Wang, and W. Yin, "Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds," in *NeurIPS*, 2018, pp. 9079–9089.
- [13] X. Xie, J. Wu, G. Liu, Z. Zhong, and Z. Lin, "Differentiable linearized admm," in *ICML*. PMLR, 2019, pp. 6902–6911.
- [14] 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*, vol. 3, no. 1, pp. 84–98, 2016.
- [15] K. Zhang, W. Zuo, and L. Zhang, "Deep plug-and-play super-resolution for arbitrary blur kernels," in *CVPR*, 2019, pp. 1671–1681.
- [16] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, "Plug-and-play image restoration with deep denoiser prior," *IEEE TPAMI*, vol. 44, no. 10, pp. 6360–6376, 2021.
- [17] J. Zhang and B. Ghanem, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," in *CVPR*, 2018, pp. 1828–1837.
- [18] Y. Yang, J. Sun, H. Li, and Z. Xu, "Admm-csnet: A deep learning approach for image compressive sensing," *IEEE TPAMI*, vol. 42, no. 3, pp. 521–538, 2018.
- [19] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 666–681, 2020.
- [20] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *ICML*. PMLR, 2019, pp. 5546–5557.
- [21] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.
- [22] A. M. Teodoro, J. M. Bioucas-Dias, and M. A. Figueiredo, "A convergent image fusion algorithm using scene-adapted gaussian-mixture-based denoising," *IEEE TIP*, vol. 28, no. 1, pp. 451–463, 2018.
- [23] Y. Sun, B. Wohlberg, and U. S. Kamilov, "An online plug-and-play algorithm for regularized image reconstruction," *IEEE Transactions on Computational Imaging*, vol. 5, no. 3, pp. 395–408, 2019.
- [24] M. Moeller, T. Mollenhoff, and D. Cremers, "Controlling neural networks via energy dissipation," in *ICCV*, 2019, pp. 3256–3265.
- [25] H. Heaton, X. Chen, Z. Wang, and W. Yin, "Safeguarded learned convex optimization," *arXiv preprint arXiv:2003.01880*, 2020.
- [26] P. Krus and J. Ölvander, "Performance index and meta-optimization of a direct search optimization method," *Engineering optimization*, vol. 45, no. 10, pp. 1167–1185, 2013.
- [27] S. Reich and A. Zaslavski, "Convergence of krasnoselskii-mann iterations of nonexpansive operators," *Mathematical and Computer Modelling*, vol. 32, no. 11–13, pp. 1423–1431, 2000.
- [28] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.
- [29] R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang, "A general descent aggregation framework for gradient-based bi-level optimization," *IEEE TPAMI*, 2022.
- [30] H. H. Bauschke, P. L. Combettes *et al.*, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011, vol. 408.
- [31] R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo, "On the iteration complexity of hypergradient computation," in *ICML*. PMLR, 2020, pp. 3748–3758.
- [32] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *ICML*, vol. 80. PMLR, 2018, pp. 1568–1577.
- [33] R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang, "A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton," in *ICML*. PMLR, 2020, pp. 6305–6315.
- [34] F. Pedregosa, "Hyperparameter optimization with approximate gradient," in *ICML*. PMLR, 2016, pp. 737–746.
- [35] R. Liu, J. Gao, J. Zhang, D. Meng, and Z. Lin, "Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond," *IEEE TPAMI*, 2021.
- [36] R. Liu, L. Ma, J. Zhang, X. Fan, and Z. Luo, "Retinex-inspired unrolling with cooperative prior architecture search for low-light image enhancement," in *CVPR*, 2021, pp. 10 561–10 570.
- [37] R. Liu, X. Liu, S. Zeng, J. Zhang, and Y. Zhang, "Value-function-based sequential minimization for bi-level optimization," *arXiv preprint arXiv:2110.04974*, 2021.
- [38] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.
- [39] D. Pfau and O. Vinyals, "Connecting generative adversarial networks and actor-critic methods," *arXiv preprint arXiv:1610.01945*, 2016.
- [40] T. Okuno, A. Takeda, A. Kawana, and M. Watanabe, "On lp-hyperparameter learning via bilevel nonsmooth optimization," *JMLR*, vol. 22, no. 245, pp. 1–47, 2021.
- [41] W. Yang, R. T. Tan, J. Feng, Z. Guo, S. Yan, and J. Liu, "Joint rain detection and removal from a single image with contextualized deep networks," *IEEE TPAMI*, vol. 42, no. 6, pp. 1377–1393, 2019.
- [42] H. Wang, Q. Xie, Q. Zhao, and D. Meng, "A model-driven deep neural network for single image rain removal," in *CVPR*, 2020, pp. 3103–3112.
- [43] X. Fu, J. Huang, D. Zeng, Y. Huang, X. Ding, and J. Paisley, "Removing rain from single images via a deep detail network," in *CVPR*, 2017, pp. 3855–3863.
- [44] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *CVPR*, 2019, pp. 3937–3946.

- [45] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *ICCV*. IEEE, 2011, pp. 479–486.
- [46] J. Kruse, C. Rother, and U. Schmidt, "Learning to push the limits of efficient fft-based image deconvolution," in *ICCV*, 2017, pp. 4586–4594.
- [47] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *CVPR*, 2017, pp. 3929–3938.
- [48] F. Lv, Y. Li, and F. Lu, "Attention guided low-light image enhancement with a large scale low-light simulation dataset," *International Journal of Computer Vision*, vol. 129, no. 7, pp. 2175–2193, 2021.
- [49] W. Wang, C. Wei, W. Yang, and J. Liu, "Gladnet: Low-light enhancement network with global awareness," in *FG*, 2018, pp. 751–755.
- [50] W. Chen, W. Wang, W. Yang, and J. Liu, "Deep retinex decomposition for low-light enhancement," in *BMVC*, 2018.
- [51] Y. Zhang, J. Zhang, and X. Guo, "Kindling the darkness: A practical low-light image enhancer," in *ACM MM*, 2019.
- [52] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, and R. Cong, "Zero-reference deep curve estimation for low-light image enhancement," in *CVPR*, 2020, pp. 1780–1789.
- [53] K. Xu, X. Yang, B. Yin, and R. W. Lau, "Learning to restore low-light images via decomposition-and-enhancement," in *CVPR*, 2020, pp. 2281–2290.
- [54] Y. Jiang, X. Gong, D. Liu, Y. Cheng, C. Fang, X. Shen, J. Yang, P. Zhou, and Z. Wang, "Enlightengan: Deep light enhancement without paired supervision," *IEEE TIP*, vol. 30, pp. 2340–2349, 2021.
- [55] W. Yang, S. Wang, Y. Fang, Y. Wang, and J. Liu, "From fidelity to perceptual quality: A semi-supervised approach for low-light image enhancement," in *CVPR*, 2020, pp. 3063–3072.
- [56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *NeurIPS*, vol. 27, 2014.
- [57] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.
- [58] F. Farnia and A. Ozdaglar, "Do gans always have nash equilibria?" in *ICML*, 2020.
- [59] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," *arXiv preprint arXiv:1711.05772*, 2017.
- [60] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *ICML*. PMLR, 2017, pp. 1165–1173.
- [61] A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots, "Truncated back-propagation for bilevel optimization," in *AISTATS*. PMLR, 2019, pp. 1723–1732.
- [62] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," in *NeurIPS*, vol. 32, 2019, pp. 113–124.
- [63] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *AISTATS*. PMLR, 2020, pp. 1540–1552.



**Xuan Liu** received the BSc degree in mathematics from Dalian University of Technology, in 2020. He is currently working toward the MPhil degree with the Department of Software Engineering, Dalian University of Technology. His research interests include computer vision, machine learning, and control and optimization.



**Shangzhi Zeng** received the BSc degree in mathematics and applied mathematics from Wuhan University, in 2015, the MPhil degree from Hong Kong Baptist University, in 2017, and the PhD degree from the University of Hong Kong, in 2021. He is currently a PIMS postdoctoral fellow with the Department of Mathematics and Statistics at University of Victoria. His current research interests include variational analysis and bilevel optimization.



**Jin Zhang** received the BA degree in journalism and the MPhil degree in mathematics and operational research and cybernetics from Dalian University of Technology, China, in 2007 and 2010, respectively, and the PhD degree in applied mathematics from University of Victoria, Canada, in 2015. After working with Hong Kong Baptist University for three years, he joined Southern University of Science and Technology as a tenure-track assistant professor with the Department of Mathematics and promoted to an associate professor in 2022. His broad research area is comprised of optimization, variational analysis and their applications in economics, engineering, and data science.



**Risheng Liu** (Member, IEEE) received the BSc and PhD degrees in mathematics from Dalian University of Technology, in 2007 and 2012, respectively. He was a visiting scholar with the Robotics Institute, Carnegie Mellon University, from 2010 to 2012. He served as a Hong Kong Scholar research fellow with the Hong Kong Polytechnic University from 2016 to 2017. He is currently a professor with the DUT-RU International School of Information Science & Engineering, Dalian University of Technology. His research

interests include machine learning, optimization, computer vision, and multimedia.



**Yixuan Zhang** received the BSc degree in mathematics and applied mathematics from Beijing Normal University, in 2020, and the MPhil degree from Southern University of Science and Technology, in 2022. She is currently working toward the PhD degree with the Department of Applied Mathematics, the Hong Kong Polytechnic University. Her current research interests include optimization and machine learning.