arXiv:2301.06081v3 [eess.IV] 12 Aug 2025

A Data-driven Loss Weighting Scheme across Heterogeneous Tasks for Image Denoising

Xiangyu Rui*, Xiangyong Cao†, Xile Zhao‡, Deyu Meng§, and Michael K. NG¶

Abstract. In a variational denoising model, weight in the data fidelity term plays the role of enhancing the noise-removal capability. It is profoundly correlated with noise information, while also balancing the data fidelity and regularization terms. However, the difficulty of assigning weight is expected to be substantial when the noise pattern is beyond independent identical Gaussian distribution, e.g., impulse noise, stripe noise, or a mixture of several patterns, etc. Furthermore, how to leverage weight to balance the data fidelity and regularization terms is even less evident. In this work, we propose a data-driven loss weighting (DLW) scheme to address these issues. Specifically, DLW trains a parameterized weight function (i.e., a neural network) that maps the noisy image to the weight. The training is achieved by a bilevel optimization framework, where the lower level problem is solving several denoising models with the same weight predicted by the weight function and the upper level problem minimizes the distance between the restored image and the clean image. In this way, information from both the noise and the regularization can be efficiently extracted to determine the weight function. DLW also facilitates the easy implementation of a trained weight function on denoising models. Numerical results verify the remarkable performance of DLW on improving the ability of various variational denoising models to handle different complex noise. This implies that DLW has the ability to transfer the noise knowledge at the model level to heterogeneous tasks beyond the training ones and the generalization theory underlying DLW is studied, validating its intrinsic transferability.

Key words. Complex noise removal, heterogeneous tasks, data fideliy term, data-driven methods.

MSC codes. 94A08, 68U10, 68T45

1. Introduction.

Image denoising task aims to restore the clean image $\mathcal{X} \in \mathbb{R}^M$ from the observed noisy image $\mathcal{Y} \in \mathbb{R}^M$. Over the past few decades, the variational optimization methods for image denoising have been extensively studied. Mathematically, these methods can be summarized as follows:

$$\hat{\mathcal{X}} = \arg\min_{\mathcal{X}} \ \ell(\mathcal{Y}, \mathcal{X}) + \lambda R(\mathcal{X}), \tag{1.1}$$

where $\hat{\mathcal{X}} \in \mathbb{R}^M$ is the restored image. There are two terms in the objective function. The data fidelity term $\ell(\mathcal{Y}, \mathcal{X})$ measures the discrepancy between $\mathcal{Y}$ and $\mathcal{X}$. The regularization term $R(\mathcal{X})$ makes the optimization problem well-posed. The trade-off parameter λ balances

*Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (xyrui.aca@gmail.com).

†School of Computer Science and Technology and the Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi, China (caoxiangyong@xjtu.edu.cn).

‡School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan, China (xlzhao122003@163.com).

§School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi, China,(dymeng@mail.xjtu.edu.cn).

¶Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (michael-ng@hkbu.edu.hk).

the data fidelity and regularization terms in the objective function. While the regularization term $R(\cdot)$ characterizes the clean image structure, the data fidelity term $\ell(\cdot,\cdot)$ implies how we characterize the noise information.

The incorporation of weight into data fidelity term can improve the noise-removal capability of a denoising model. To clarify the functional principles of weight, we can concretized model (1.1) as follows:

$$\hat{\mathcal{X}} = \arg\min_{\mathcal{X}} \underbrace{\frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{X})\|^2}_{\ell(\mathcal{Y},\mathcal{X};\mathcal{W})} + R(\mathcal{X}), \tag{1.2}$$

where "$\|\cdot\|$" represents the ℓ_2 norm and "$\odot$" represents the Hadmard product, and elements of the weight $\mathcal{W}$ are non-negative. In the data fidility term, each element of $\mathcal{W}$ determines how much the element of $\mathcal{Y}$ in the same position can contribute to the problem solution. For example, suppose an area of noisy image $\mathcal{Y}$, e.g., $\mathcal{Y}(\Omega)$, is more heavily polluted than the rest area of $\mathcal{Y}$, this area Ω is then not "trustworthy" for construct $\mathcal{X}$. Thus, it is reasonable to set the value of $\mathcal{W}(\Omega)$ in model (1.2) to be smaller, implying that $(\mathcal{Y}-\mathcal{X})(\Omega)$ is not supposed to be too small. From this toy example we see that a reasonable $\mathcal{W}$ and noise information are deeply correlated. However, determining the value of $\mathcal{W}$ according to noise information could be rather challenging when the noise pattern is complex, e.g., impluse noise, stripe noise, or a mixture of several patterns. One shortcut approach is to set all the values of $\mathcal{W}$ to be one. In this way, the data fidelity term $\ell(\mathcal{Y},\mathcal{X};1)$ does not take into consideration the variability of noise within an image, which would inevitably limit the model performance. Weight $\mathcal{W}$ also acts as a bridge connecting the data fidelity term $\ell(\mathcal{Y},\mathcal{X};\mathcal{W})$ and the regularization term $R(\mathcal{X})$. It changes with $R(\mathcal{X})$ and our expectation on the solution $\hat{\mathcal{X}}$. A toy example is that $\mathcal{W}$ should increase $\sqrt{\eta}$-fold if $R(\mathcal{X})$ increases η-fold and the solution keeps unchanged. Compared with scalar trade-off parameter λ in (1.1), $\mathcal{W}$ establishes the connection in a finer-grained way because it operates on each element, which, however, is more implicit and result in greater modeling challenges.

Computing a non-trivial $\mathcal{W}$ in existing image denoising methods mainly involve the following forms. a) $\mathcal{W}$ is set by an empirical formula which roughly follows the rule that $\mathcal{W}$ should be inversely proportional to the estimated noise intensity "$|\mathcal{Y}-\mathcal{X}|$" [41]. b) Taking model (1.2) as the maximum a posteriori (MAP) problem, $\mathcal{W}$ is obtained from the negtive loglikelihood "$-\log p(\mathcal{Y}|\mathcal{X})$". In this case, the noise is assumed to take specific forms, e.g., Gaussian noise [43], poisson and mixed poisson-Gaussian noise [21] and Gaussian mixture noise [22]. The parameters of the assumed noise distribution are known or estimated to calculate $\mathcal{W}$. c) $\mathcal{W}$ is also considered to be parameters of a hypothetical noise distribution. It is calculated and iteratively updated via inference algorithms such as Expectation-Maximization (EM) [8, 16] and variational inference [11, 46]. In summary, the weighting strategies in existing methods mainly depend on empirical formulation or specific hypothetical noise distribution. As the noise pattern becomes more complex, the implementation difficulty of these methods increases accordingly. Besides, the regularization term is usually left out and only the noise information is considered, which would also limit the performance.

To address the issue, we consider to predict the weight $\mathcal{W}$ in the data fidelity term $\frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{X})\|^2$ automatically by a parametrized "weight function". In this work, we propose a

data-driven loss weight (DLW) scheme to train a "weight function" which is a lightweight neural network "h_θ" (called DLWnet for short). The DLWnet takes the noisy image, which contains all image and noise information, as input and directly generates the weight as output. Unlike all previous works, we consider completely leaving the denoising models of the form (1.2) themselves and data to determine this weight function. There are two basic but very important facts to help to realize such idea:

- Different weight $\mathcal{W}$s would produce different solutions in one model (1.2) and thus the best solution should imply the most suitable weight.
- Different regularizations $R(\cdot)$ with the same weight would also produce different restored images $\hat{\mathcal{X}}$.

Against such issues, in this study we describe the implicit relationship between the restored image and its proper weight function by a bilevel optimization framework given noisy/clean image training pairs. Specifically, such a bilevel framework is composed of two sub-problems. The lower-level subproblem leverages multiple denoising models of form (1.2) to generate the restored images by optimizing these models. Each of them contains the same weight $\mathcal{W}$ predicted by an explicit weighting function h_θ mapping from an image to its imposed weight setting. The upper-level sub-problem aims to optimize the neural network parameters θ of h_θ by minimizing the discrepancy between the restored and corresponding ground-truth images. The above bi-level optimization process finishes the training of h_θ. We see that h_θ is essentially guided to learn to extract the noise information and features of regularization from models and training pairs with very specific physical meaning. Then, h_θ can be readily used to estimate the weight of other unseen noisy images as well as used for other regularizations. Subsequently, the predicted weight help to enhance the performance of a denoising model (1.2) to adapt to complex noise.

Note that denoising models with remarkable performance always have complicated regularizations and may not be able to train h_θ in the bilevel framework. Thus, we consider to use several denoising models with simpler regularizations (namely, source models) for training h_θ. This would raise another question. What effect does the choice of the source models have on the performance of a to-be-used denoising model (namely, the target model)? To analyze such generalization ability of h_θ at the model level, we have conducted several experiments and observe their results. Besides, we have also preliminarily established a theoretical analysis about the generalization error from the source models to the target model by the proposed model divergece, which helps to intrinsically understand the generalization behavior of h_θ across heterogeneous image denoising models.

In summary, the contribution of this study is as follows:

1. We propose a data-driven loss weighting (DLW) scheme to learn the weight function h_θ that calculates the weight of the data fidelity term in an variational denoising model for image denoising. The proposed scheme does not require any empirical or hypothetical assumptions on noise, and leaves denoising models to determine the weight function.
2. The learned h_θ by the proposed DLW scheme can directly predict weights for new noisy images and the weight can help to improve the complex noise removal ability of the denoising model. Extensive experiments demonstrate the capability of our DLWnet to handle various complex noises.
3. The learned h_θ is ready to be used for a new image denoising model with different

regularization terms. Substantial experiments about the source model combinations and their effects on heterogeneous target models show that h_θ has good generalization ability at the model level.

4. Some preliminary theories are established to analyze the generalization error of DLW at the model level. The theories help to understand how the choice of source models would have on the target model.

The rest of this work is organized as follows. Sec. 2 reviews weighting schemes in existing image denoising models and presents relevant learning theory works. Sec. 3 presents our DLW scheme to predict the weight and disscussions about the weight. Additionally, the theoretical analysis of generalization error of our method is provided. Sec. 4 demonstrates experiments to verify the generalization ability of our DLWnet both on different noise patterns and different regularizations from training ones. The paper is then concluded.

2. Related Works.

2.1. Weighting sheme for data-fidelity term. Implementing weight in the data-fidelity for variational image denoising models has gain more attentions in recent years. The aim of using weight is to leverage more noise information such that the denoising models could have better performance. An intuitive weighting scheme is to let the weight be inversely proportional to the estimated noise intensity $|\mathcal{Y} - \mathcal{X}|$. For example, Wang et al. [41] set the weight as $\exp(-\xi(\mathcal{Y}-\mathcal{X})^2)$ and update the weight in each iteration when solving the denoising model. However, this sheme fails to capture the intrinsic characteristics of the weight.

A more interpretable approach involves establishing a correlation between model (1.2) and MAP to derive the weight values from the likelihood. Consequently, this method requires to make assumptions on the noise distribution. Liu et at. [22] assumes that the noise distribution is the mixture of several distributions. In practice, they use Gaussian mixture noise and the weight is derived from the corresponding log-likehood. Xu et al. [43] presents a multi-channel WNNM model. The noise in each channel follows a Gaussian distribution, but the variances differ across channels. The weights on a single channel are the same and is set as the noise precision. Chen et al. [10] enhanced this method by set weight as the combination of noise precision and the identity matrix. Liu et al. [21] assumes that the image is corrupted with Poisson noise and use additive Gaussian noise to approximate the Poisson noise. The weight is derived from the negative log of the Gaussian distribution with the denoised image as the variation. Similarly, from a probabilistic modeling perspective, $\mathcal{W}$ can also be considered as parameters of a hypothetical noise distribution and updated during EM or variational inference. Cao et al. [8] models noise with the mixture of exponential power and use EM to solve the denoising model. Jiang et al. [16] assumes that the noise follows Gaussian mixture distribution and use EM to solve the corresponding MAP problem. Chen et al. [11] also assumes that the noise follows Gaussian mixture distribution but the problem is related to variational inference and the weight is updated by the corresponding rule. There are also special cases for computing weight for image denoising models. Li et al. [20] considers the Aubert–Aujol model for removing multiplicative noise that is assumed to follow the Gamma distribution. The weights in the data-fidelity term also operate elementwisely. They are iteratively updated via the rule derived from Euler–Lagrange equation from energy function.

The proposed DLW scheme is completely different from the above weighting schemes.

We neither employ empirical formulas to set weight nor make assumptions about the noise distribution. DLW is a data-driven approach that learns a parameterized mapping to calculate weight. Our method is not constrained to specific noise patterns or specific denoising models and is also easy to be implemented into a target denoising model.

2.2. Learning Theory on Generalization. In recent decades, the learning theory on generalization has gained great development in transfer learning [13, 24], domain adaption [4, 12, 14, 27], domain generalization [6, 29, 5, 37], and so on. For example, [26, 39] considered the generalization theory of the method that learns the common representation from several training tasks and generalizes to new tasks from the perspective of task diversity. [35] studied the generalization theory of the approach that learns a universal hyper-parameter prediction policy for a group of tasks. The generalization theories of these works are developed mainly based on a pre-assumed common environment in which all tasks are drawn. Additionally, the diversity of the environment should be bounded by the diversity of all training tasks so that the generalization makes sense.

In this study, inspired by these related generalization researches, we prove the generalization theory of our DLW for image denoising models. The main differences between our method and other generalization theories are as follows. Firstly, we turn to measure the individual task divergence rather than the environment, which is more pertinent to our problem. Secondly, the existing generalization theory works mainly focus on classification problems, while our work should be the first generalization theory work in the field of image denoising to the best of our knowledge.

3. Proposed Method. As mentioned in the introduction section, we want to break free from the empirical formulation or hypothetical noise distribution when designing weight. To address the issue, a parameterized function is utilized to build a mapping from the noisy image $\mathcal{Y}$ to its weight $\mathcal{W}$ which is supposed to reflect noise information:

$$\mathcal{W} = h_\theta(\mathcal{Y}), \tag{3.1}$$

where $h_\theta(\cdot)$ is a lightweight neural network with parameter θ. The DLWnet h_θ receives the noisy image $\mathcal{Y}$ as the input and generates the weight $\mathcal{W}$ as the output. Correspondingly, the data fidelity term $\frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{X})\|^2$ can be rewritten as $\frac{1}{2}\|h_\theta(\mathcal{Y})\odot(\mathcal{Y}-\mathcal{X})\|^2$. In this section, we will introduce our DLW scheme to derive the weight function h_θ for image denoising model, including its training and practical application, which is shown in Figure 1.

3.1. Notations. We use $\mathcal{X}/\mathcal{Y}$ and $\mathcal{X}\odot\mathcal{Y}$ to represent the element-wise division and multiplication between $\mathcal{X}$ and $\mathcal{Y}$, respectively. Unless otherwise specified, the square operator $\mathcal{X}^2$ in this work also denotes element-wise square. The i-th element of $\mathcal{X}$ is writen as $\mathcal{X}(i)$. The norm $\|\mathcal{X}\|$ means Frobenius norm of $\mathcal{X}$, i.e., $\|X\| = \sqrt{\sum_{i=1}^{M}\mathcal{X}(i)^2}$. The inner product between $\mathcal{X}$ and $\mathcal{Y}$ is denoted as $\langle\mathcal{X},\mathcal{Y}\rangle$. The set $\{1,2,...,N\}$ is writen as $[N]$ for short.

3.2. Training DLWnet h_θ from multiple source models. This section introduces the proposed DLW scheme. Combining (3.1), we can rewrite problem (1.2) as

$$\hat{\mathcal{X}}(\mathcal{Y},\theta,R) = \arg\min_{\mathcal{X}} \frac{1}{2}\|h_\theta(\mathcal{Y})\odot(\mathcal{Y}-\mathcal{X})\|^2 + R(\mathcal{X}). \tag{3.2}$$

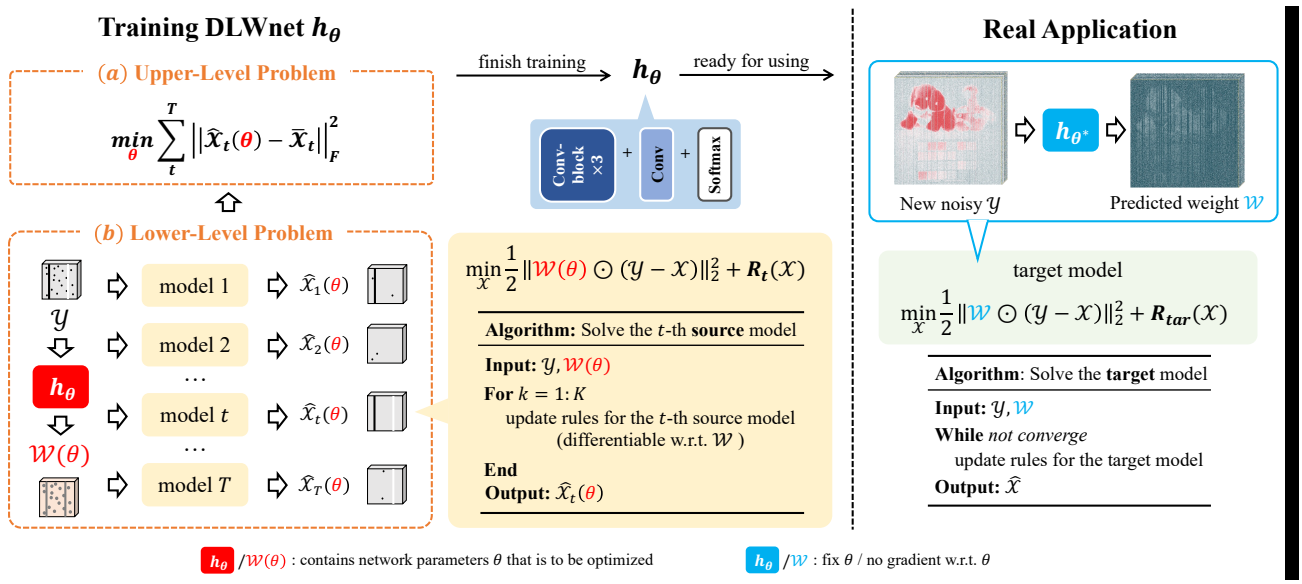


Figure 1: Overview of the proposed DLW scheme. The left part shows the training process of the weight function h_θ (DLWnet). This process contains a lower-level problem that solves T heterogeneous image denoising problems which are implemented with the same h_θ, and an upper-level problem that minimizes the distance between the resotred image $\hat{\mathcal{X}}(\theta)$ and the ground-truth image $\bar{\boldsymbol{\mathcal{X}}}$. The right part shows the application of DLWnet h_θ. In this stage, a trained h_θ predicts the weight of a noisy image for a target image denoising model to help the model better handling complex noise and achieve better performance.

Here, the solution to the optimization problem (1.2), i.e., the restored image $\hat{\mathcal{X}}$, has a mapping relation with three elements. Without a doubt, the noisy image $\mathcal{Y}$ is the basic element to find $\hat{\mathcal{X}}$. The second element is the network parameter θ. It first determines the weight $\mathcal{W} = h_\theta(\mathcal{Y})$ and then the weight influences the optimization results. The third element is the regularization term R which distinguishes the denoising models. Different denoising models would also result in different restored images. Overall, the noisy image $\mathcal{Y}$, the network parameter θ and the model regularization R collectively determine the final restored image $\hat{\mathcal{X}}$. Fig. 2 briefly shows their relationship. Note that such mapping relation is implicit because it is achieved by solving an optimization problem.

From (3.2), it is clear that to train network h_θ, we need "clean/noisy" training data and denoising models. Thus, a bi-level optimization framework is formulated as follows:

$$\begin{cases} \min_\theta \ \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} \dfrac{1}{N_t} \sum_{i_t=1}^{N_t} \ell_{up}\left(\hat{\mathcal{X}}_{t,i_t}(\theta), \bar{\boldsymbol{\mathcal{X}}}_{t,i_t}\right), & \text{(3.3a)} \\ \hat{\mathcal{X}}_{t,i_t}(\theta) = \arg\min_{\mathcal{X}} \dfrac{1}{2}\|h_\theta(\mathcal{Y}_{t,i_t}) \odot (\mathcal{Y}_{t,i_t} - \mathcal{X})\|^2 + R_t(\mathcal{X}), \ (t = 1, ..., T). & \text{(3.3b)} \end{cases}$$

Problem (3.3a) and (3.3a) are called upper-level and lower-level problems, respectively. There

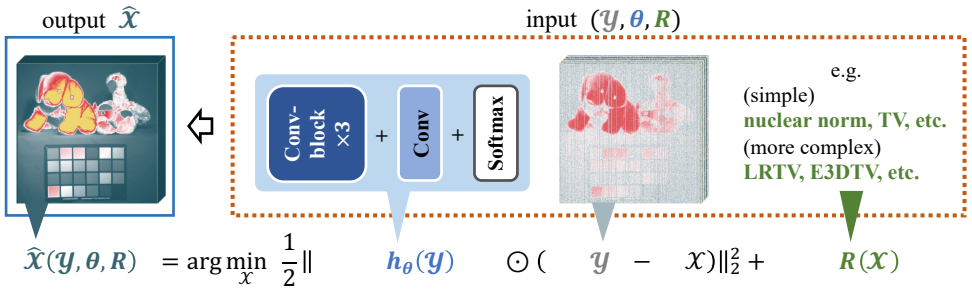


Figure 2: The solution $\hat{\mathcal{X}}$ to a denoising problem is implicitly related three elements. They are the noisy image $\mathcal{Y}$, the network parameter θ (because $\mathcal{W} = h_\theta(\mathcal{Y})$) and the regularization term $R(\mathcal{X})$. This fact is the basis of the proposed DLW scheme.

are T different denoising models in the lower-level problem (3.3a). $R_t(\mathcal{X})$ is the regularization term of the t-th denoising model. The training dataset is written as $\{(\mathcal{Y}_{t,i_t}, \bar{\boldsymbol{\mathcal{X}}}_{t,i_t})\}_{i_t=1}^{N_t}{}_{t=1}^{T}$, where $\mathcal{Y}_{t,i_t}$ means the i_t-th noisy image for the t-th denoising model and $\bar{\boldsymbol{\mathcal{X}}}_{t,i_t}$ is the corresponding ground-truth image. N_t is the number of training pairs for the t-th model. For convenience, $\hat{\mathcal{X}}_{t,i_t}(\theta)$ is the abbreviation of $\hat{\mathcal{X}}(\mathcal{Y}_{t,i_t}, \theta, R_t)$, which means the solution to the t-th denoising model with noisy image $\mathcal{Y}_{t,i_t}$.

It should be indicated that training h_θ using the proposed bi-level optimization framework (3.3a)-(3.3a) does not rely on extra knowledge about weight $\mathcal{W}$. It completely allows the denoising models to decide the weight function h_θ because the output of h_θ serves the same role as the general weights in the denoising model. Besides, the lower-level problem (3.3a) produces the restored images from different denoising models using the same weight function h_θ. Thus, h_θ is able to integrate noise information provided by multiple models. This should be important because h_θ trained by a single denoising model would inevitably be more advantageous for that model. Then, in the upper-level problem (3.3a), the network parameter θ is optimized by minimizing the distance between $\hat{\mathcal{X}}$ and $\bar{\boldsymbol{\mathcal{X}}}$, the former of which carries the "weighting rule" from h_θ.

To solve the bilevel optimization problem (3.3a)-(3.3a), we utilize an efficient method called "unrolling". The idea of unrolling is applying an algorithm to solve the lower problem with a fixed number of steps of iteration. The entire process of iteration is in place of the lower level problem. Consequently, the algorithm output serves as an approximation to the real solution $\hat{\mathcal{X}}(\theta)$. For example, in Algorithm 3.1, we show the general form of applying Alternating Direction Method of Multipliers (ADMM) [7] to solve model (1.2). There are two points to concern for the lower-level problem. First, we need to set an appropriate number of steps to run the algorithm because it usually takes many steps for the algorithm to converge while the computational graph of each step has to be stored. An excessive number of steps not only escalates computational resource requirements but also compromises computational efficiency. Second, each updating rule in the algorithm should be differentiable with respect to (w.r.t.) their variables such that the final $\hat{\mathcal{X}}$ is differentiable w.r.t. $\mathcal{W}$ and gradient descent based methods can be used for the upper-level problem. Thus, we tend to choose simple denoising

Algorithm 3.1 An example of ADMM algorithm to solve denoising problem (1.2)

Input: noisy image $\mathcal{Y}$, weight $\mathcal{W}$.
Initialization: $\mathcal{X}, \mathcal{Z}, \Gamma, \mu$
while not convergence **do**
  $\mathcal{Z} \leftarrow \dfrac{\mathcal{W}^2 \odot \mathcal{Y} + \mu\mathcal{X} + \Gamma}{\mathcal{W}^2 + \mu}$
  $\mathcal{X} \leftarrow \arg\min_{\mathcal{X}} \dfrac{\mu}{2} \left\| \mathcal{X} - (\mathcal{Z} - \dfrac{\Gamma}{\mu}) \right\|^2 + R(\mathcal{X})$
  $\Gamma \leftarrow \Gamma + \mu(\mathcal{X} - \mathcal{Z})$
end while
Output: $\hat{\mathcal{X}} = \mathcal{X}$

models (source models) when designing the lower-level problem for more efficient training. For example, denoising models with nuclear norm or total variation norm as regularization usually have closed-form solutions for the corresponding $\mathcal{X}$-subproblem (update $\mathcal{X}$) and are relatively easy to optimize. For the upper level problem (3.3a), general methods for optimizing neural networks can be used, e.g., ADAM [18]. By chain rule, the gradients w.r.t. θ backpropagate through the iteration process. Specifically, let $\mathcal{X}_{k+1} = g_{k+1}[\mathcal{X}_k; h_\theta(\mathcal{Y})]$ denote the $(k+1)$-th step of iteration and the total number of steps is K. The lower level problem then is replaced by

$$\hat{\mathcal{X}}_K(\theta) = g_K[g_{K-1}[...g_1[\mathcal{X}_0; h_\theta(\mathcal{Y})]; h_\theta(\mathcal{Y})]; h_\theta(\mathcal{Y})] \tag{3.4}$$

Let $H_k^{(\mathcal{X})} = \frac{\nabla g_k}{\nabla \mathcal{X}_{k-1}}$ and $H_k^{(\mathcal{W})} = \frac{\nabla g_k}{\nabla h_\theta(\mathcal{Y})} \frac{\nabla h_\theta(\mathcal{Y})}{\nabla \theta}$. The gradient w.r.t. network parameter θ can be roughly written as

$$\frac{\nabla \ell_{up}(\hat{\mathcal{X}}_K(\theta), \bar{\boldsymbol{\mathcal{X}}})}{\nabla \theta} = \frac{\nabla \ell_{up}(\hat{\mathcal{X}}_K, \bar{\boldsymbol{\mathcal{X}}})}{\nabla \hat{\mathcal{X}}_K} \underbrace{\left[H_K^{(\mathcal{X})} \left[H_{K-1}^{(\mathcal{X})} \left[...H_1^{(\mathcal{X})} + H_1^{(\mathcal{W})}\right] + H_{K-1}^{(\mathcal{W})}\right] + H_K^{(\mathcal{W})}\right]}_{\text{through iteration}} \tag{3.5}$$

In Algorithm 3.2, we summarize the entire training process of our DLW scheme.

3.3. Applications of DLWnet h_θ. In the training phase, h_θ learns how to extract noise information from a noisy image and generate the corresponding weight. The weight should be used for a determinate image denoising model to better remove complex noise. The implementation of a trained h_θ is straightforward. Specifically, the data fidelity term $\|h_\theta(\mathcal{Y}) \odot (\mathcal{Y} - \mathcal{X})\|^2$ is plugged into a target image denoising model. The consequent general form of application of h_θ is as follows:

$$\begin{aligned} &\min_{\mathcal{X}} \ \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{X})\|^2 + \tau R_{tar}(\mathcal{X}), \\ &\text{where } \mathcal{W} = h_\theta(\mathcal{Y}). \end{aligned} \tag{3.6}$$

Note that the target model does not necessarily have to belong to the source models. If the goal is to achieve better performance, denoising models usually need to be carefully designed

Algorithm 3.2 Solve the bi-level problem (3.3a)-(3.3a)

Require: training dataset $\{(\mathcal{Y}_{t,i_t}, \bar{\boldsymbol{\mathcal{X}}}_{t,i_t})\}_{i_t=1}^{N_t}{}_{t=1}^{T}$, untrained network h_θ, T denoising models of form (1.2), number of total training steps J.
Initialization: θ_0
for $j = 1 : J$ **do**
  for $t = 1 : T$ **do**
    for $i_t = 1 : N_t$ **do**
      predict weight $\mathcal{W}_{t,i_t} = h_\theta(\mathcal{Y}_{t,i_t})$
      calculate $\hat{\mathcal{X}}_K(\theta)$ by (3.4)
    end for
  end for
  calculate gradient $\nabla\theta$ by (3.5)
  update θ_j by a forward step of network training
end for
Output: h_θ

and the regularizations are complex. However, training convenience and efficiency are the main principles of selecting source models. The denoising problem (3.6) is solved as general optimization problems using suitable algorithm, e.g., proximal gradient (PG) [34], half quadratic splitting (HQS) [38] and ADMM [7], etc.

There are also some special cases of problem (3.6). For example, when the deep image prior (DIP) [40, 19, 23] is implemented as image regularization, problem (3.6) can be modified into the following form:

$$\begin{aligned} \min_{\eta} \ & \|\mathcal{W} \odot (g_\eta(\mathcal{Z}) - \mathcal{Y})\|^2 + \tau R(g_\eta(\mathcal{Z})), \\ & \text{where } \mathcal{W} = h_\theta(\mathcal{Y}). \end{aligned} \tag{3.7}$$

In problem (3.7), $g_\eta(\cdot)$ is a deep neural network (DNN) that takes random noise $\mathcal{Z}$ as input and generates the denoised image as output. Difference between general model (3.6) and DIP model (3.7) lies on that the structure of $g_\eta(\cdot)$ also provides implicit image regularization. Once the training is finished, i.e., the optimal parameter η^* is learned, the restored image $\hat{\mathcal{X}}$ can be immediately obtained as $g_{\eta^*}(\mathcal{Z})$.

Compared with previous methods, our h_θ can directly generate the proper weight and the underlying weighting rules are learned from data and models. It should also be noted that our h_θ is not designed for one specific model. It can be easily implemented to different regularizations with different regularization terms.

3.4. Theoratical analysis. When the source model is the same as the target model, we get the ideal h_θ for the target model. On the contrary, there exists "error" when we apply h_θ if the source and the target models are different. Specifically, such "error" can be described as the difference between the average performance of the utilized h_θ and the ideal h_θ implementing on the target model. We use "$\mathbf{E}_g$" to refer to the error, which is also called the generalization error. In this section, we will present a preliminary estimation of the upper bound related to

the generalization error.

More generally speaking, suppose there are S target models of the form in problem (1.2). They are written as

$$\min_{\mathcal{X}} \frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{X})\|^2+R_t(\mathcal{X}),\ t=T+1,...,T+S, \tag{3.8}$$

For convenience, the subscript character used in (3.3a) for T source models continues to be used for the S target models, while the index for target models start counting from $T+1$ and ends at $T+S$. Then our main result about the error $\mathbf{E}_g$ brougnt by transfering a learned h_θ to target models is presented in the following theorem.

Let data space $\mathbb{Y}$ and $\mathbb{X}$ denote the observed image space and the original image space, respectively. We have that $\mathcal{Y}\in\mathbb{Y}$ and $\bar{\boldsymbol{\mathcal{X}}}\in\mathbb{X}$. Let D represent a joint distribution defined on $\mathbb{Y}\times\mathbb{X}$. To be specific, the joint probability density function (PDF) $p(\mathcal{Y},\mathcal{X})$ is usually decomposed as

$$p(\mathcal{Y},\mathcal{X})=p(\mathcal{Y}|\mathcal{X})p(\mathcal{X}),$$

where $p(\mathcal{X})$ represents the PDF of the original data $\mathcal{X}$, and $p(\mathcal{Y}|\mathcal{X})$ reveals the generative process of the observed data $\mathcal{Y}$ from $\mathcal{X}$.

We firstly present a Lemma that guarantees the existence of implicit mapping from $\mathcal{Y}$, $\mathcal{W}$ and R to $\hat{\mathcal{X}}$ in model (1.2). The proof is presented in Appendix A.

Lemma 3.1. *Suppose $\mathcal{W}\in[\varepsilon,B_w]^M$, where $0<\varepsilon\le B_w$, and R is proper, closed and convex. Then the solution to model (1.2) is unique.*

We assume that the observed image space $\mathbb{Y}$ and original image space $\mathbb{X}$ are bounded in a cube in $\mathbb{R}^M$, i.e., $\mathbb{Y},\mathbb{X}\subset[-B_d,B_d]^M$ for a positive constant B_d. Note that the boundedness of $\mathcal{Y}$ implies that $\hat{\mathcal{X}}$ is also bounded. And we assume that $\hat{\mathcal{X}}\in[-B_d,B_d]^M$ for convenience.

In the following of this work, we also use $f_R^h(\mathcal{Y})$ to denote the restored image $\hat{\mathcal{X}}(\mathcal{Y},\theta,R)$ and continue to call "f_R^h" a source model. The T source models $\{f_{R_t}^h\}_{t=1}^T$ in the lower level problem (3.3a) can be aggregated into a set:

$$\mathcal{F}_{\mathcal{H}}:=\{(f_{R_1}^h,...,f_{R_T}^h)|h\in\mathcal{H}\}.$$

Each element $(f_{R_1}^h,...,f_{R_T}^h)$ in $\mathcal{F}_{\mathcal{H}}$ contains T source models with the same network h. The elements are distinguished from each other by the choice of network h. The networks in $\mathcal{H}$ have the same structure but different network parameters.

3.4.1. Training Error Estimation. The objective function of upper level problem (3.3a) can be equivalently written as

$$\min_{h\in\mathcal{H}}\ \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\ell_{up}\left(f_{R_t}^h(\mathcal{Y}_{t,i_t}),\bar{\boldsymbol{\mathcal{X}}}_{t,i_t}\right), \tag{3.9}$$

which is the *empirical risk minimization* (ERM) problem [28] of $h\in\mathcal{H}$ on training dataset $\mathbf{S}=\{(\mathcal{Y}_{t,i_t},\bar{\boldsymbol{\mathcal{X}}}_{t,i_t})\}_{i=1\,t=1}^{N_t\ T}$. Therefore, our training loss, or empirical risk, refers to

$$\hat{\mathbf{R}}_{tr}(h):=\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\ell_{up}\left(f_{R_t}^h(\mathcal{Y}_{t,i_t}),\bar{\boldsymbol{\mathcal{X}}}_{t,i_t}\right).$$

And the expected risk is

$$\mathbf{R}_{tr}(h) := \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left[\ell_{up}\left(f_{R_t}^{h}(\mathcal{Y}),\mathcal{X}\right)\right].$$

The expected risk takes expectations over joint distribution D. Correspondingly, the "best" DLWnets are defined as

$$\hat{h}_{tr} := \arg\min_{h\in\mathcal{H}} \hat{\mathbf{R}}_{tr}(h),\ \ h_{tr}^{*} := \arg\min_{h\in\mathcal{H}} \mathbf{R}_{tr}(h),$$

where $\hat{h}_{tr}$ is the final trained DLWnet under the bi-level framework (3.3a)-(3.3a) with a finite number of training samples , and h_{tr}^{*} is the ideal DLWnet which is calculated from the underlying sample distribution D. Since the expected h_{tr}^{*} is intractable, the training error $\mathbf{E}_{tr}$ to measure the "closeness" between $\hat{h}_{tr}$ and h_{tr}^{*} is then defined as:

$$\mathbf{E}_{tr} := \mathbf{R}_{tr}(\hat{h}_{tr}) - \mathbf{R}_{tr}(h_{tr}^{*}). \tag{3.10}$$

Next, we estimate the upper bound of $\mathbf{E}_{tr}$. To achieve this goal, we first need to present some mild assumptions on $\mathcal{H}$ and ℓ_{up} and properties of f_R^h.

Assumption 3.2. $\forall h\in\mathcal{H},\mathcal{Y}\in\mathbb{Y}$, we have $h(\mathcal{Y})\in[\varepsilon,B_w]^M$,, where $0<\varepsilon\le B_w$.

Assumption 3.3. ℓ_{up} satisfies $|\ell_{up}(\hat{\mathcal{X}},\mathcal{X})|\le B_l$ for all $\hat{\mathcal{X}},\mathcal{X}\in[-B_d,B_d]^M$. And $\ell_{up}(\hat{\mathcal{X}},\mathcal{X})$ is L_n-Lipschitz w.r.t. $\hat{\mathcal{X}}$ for all $\mathcal{X}\in[-B_d,B_d]^M$.

Remark 3.4. The mean squared error (MSE) $\frac{1}{M}||\hat{\mathcal{X}}-\mathcal{X}||^2$ is bounded by $4B_d^2$ and is $\frac{4B_d}{\sqrt{M}}$-Lipschitz with respect to $\hat{\mathcal{X}}\in[-B_d,B_d]^M$ for all $\mathcal{X}\in[-B_d,B_d]^M$.

Lemma 3.5. *Suppose that R be proper, convex and closed, and Assumption 3.2 hold. Then $\forall\mathcal{Y}\in\mathbb{Y}$ and $\forall h_1,h_2\in\mathcal{H}$, we have*

$$\left\|f_R^{h_1}(\mathcal{Y})-f_R^{h_2}(\mathcal{Y})\right\|\le L_H\left\|h_1(\mathcal{Y})-h_2(\mathcal{Y})\right\|. \tag{3.11}$$

The Lipschitz constant L_H only depends on B_d, ε and B_w.

Lemma 3.5 shows the Lipschitz continuity of f_R^h w.r.t. weight h, which means that if two weights $h_1(\mathcal{Y})$ and $h_2(\mathcal{Y})$ are close enough, the corresponding restored images $f_R^{h_1}(\mathcal{Y})$ and $f_R^{h_2}(\mathcal{Y})$ are also close enough.

We use Gaussian complexity captures the richness of a set of functions by measuring the degree to which the functions in this set can fit random noise.

Definition 3.6 (Empirical Gaussian complexity [2]). *Let $\mathcal{F}$ be a set of functions mapping from $\mathcal{Z}$ to [a,b] and $S=(z_1,...,z_m)$ a fixed sample of size m with elements in $\mathcal{Z}$. Then, the empirical Gaussian complexity of $\mathcal{F}$ with respect to the sample S is defined as:*

$$\hat{\mathfrak{G}}_S(\mathcal{F}) := \mathop{\mathbb{E}}_{\boldsymbol{g}}\left[\sup_{f\in\mathcal{F}}\frac{1}{m}\sum_{i=1}^{m}g_i f(z_i)\right],\ \ g_i\sim\mathcal{N}(0,1),$$

where $\boldsymbol{g}=(g_1,...,g_m)^T$.

The empirical Gaussian complexity of $\mathcal{F}_{\mathcal{H}}$ is then defined as

$$\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}}) := \mathop{\mathbb{E}}_{\boldsymbol{G}}\left[\sup_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M} g_{t,i_t,m} f^h_{R_t}(\mathcal{Y}_{t,i_t})_m\right],\ g_{t,i_t,m}\sim\mathcal{N}(0,1),$$

where $\boldsymbol{G} = (g_{1,1,1},...,g_{t,i_t,m},...,g_{T,N_T,M})^T$ and $f^h_{R_t}(\cdot)_m$ is the m-th element of $f^h_{R_t}(\cdot)$.

Based on Assumption 3.2 and Lemma 3.5, we can decouple $\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}})$ using $\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{H})$ via an analogous chain rule [2]. Now, we present our main conclusion about the training error $\mathbf{E}_{tr}$.

Theorem 3.7 (Training error). *Suppose that Assumption 3.2 and Assumption 3.3 hold. $\{R_t\}_{t=1}^T$ are proper, closed and convex. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following inequality hold*

$$\mathbf{E}_{tr} \le 6L_nL_H\hat{\mathfrak{G}}_S(\mathcal{H}) + \frac{6B_l}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}}. \tag{3.12}$$

The first term of (3.12) is about the Gaussian complexity of $\mathcal{H}$. It naturally implies that the more complex $\mathcal{H}$ is, the more uncertainty the training process would bring in about $\hat{h}_{tr}$. The second term of (3.12) is mainly about δ and the size of the training data set $\mathbf{S}$. If we raise the value of $1-\delta$ to make (3.12) more probable, the value of $\sqrt{\frac{1}{2}\log\frac{2}{\delta}}$ would be higher. Then, to lower the second term, we should enlarge the size of $\mathbf{S}$, i.e. collect more training data. Besides, if one source model lacks training samples, the insufficient training of h brought by this task would degrade the joint training performance, even if we increase the training samples for other source models. This could also be quantified by seeing that $\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}$ is the aggregation of $1/N_t$ for all $t\in[T]$. Therefore, we can see that each source model contributes to the final training performance.

3.4.2. Generalization Error Estimation. Similar to the training error, we establish the test error for S target models (3.8) as:

$$\mathbf{R}_{te}(h) := \frac{1}{S}\sum_{t=T+1}^{T+S}\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim\mathrm{D}}\left[\ell_{up}\left(f^h_{R_t}(\mathcal{Y}),\mathcal{X}\right)\right],$$

where $\{R_t\}_{t=T+1}^{T+S}$ are usually different from $\{R_t\}_{t=1}^{T}$. The corresponding "best" DLWnet for the target models can then be defined as:

$$h_0^* = \arg\min_{h\in\mathcal{H}}\mathbf{R}_{te}(h).$$

The generalization error is thus defined as:

$$\mathbf{E}_g := \mathbf{R}_{te}(\hat{h}_{tr}) - \mathbf{R}_{te}(h_0^*).$$

Specifically, $\mathbf{R}_{te}(\hat{h}_{tr})$ represents the test error on the trained DLWnet $\hat{h}_{tr}$ and $\mathbf{R}_{te}(h_0^*)$ is the test error infimum on the set $\mathcal{H}$. To estimate $\mathbf{E}_g$, our main challenge relies on measuring the divergence between the source and target tasks, i.e., calculating $\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h)$.

The major difficulty of calculating $\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h)$ comes from the implicit form of f_R^h. We thus seek an approximation of f_R^h to ease the calculation by means of the modified gradient step [30]. Specifically, by taking the derivative of the objective function of (1.2) with respect to $\mathcal{X}$, the solution $\hat{\mathcal{X}}$ should satisfy:

$$\begin{aligned} &\mathcal{W}^2 \odot (\hat{\mathcal{X}} - \mathcal{Y}) + \nabla R(\hat{\mathcal{X}}) = 0, \\ \Rightarrow &\hat{\mathcal{X}} = \left(\mathbf{I} + \frac{1}{\mathcal{W}^2} \odot \nabla R\right)^{-1} (\mathcal{Y}), \end{aligned} \tag{3.13}$$

where $\mathbf{I}$ represents the identity mapping. The operator $(\mathbf{I} + g)^{-1}$ is known as the resolvent operator [42]. If g is nonlinear, its resolvent form is still implicit and difficult to analyze. Therefore, we approximate Eq. (3.13) by using a linear transform to ∇R[30][42]:

$$\left(\mathbf{I} + \frac{1}{\mathcal{W}^2} \odot \nabla R\right)^{-1} \approx \mathbf{I} - \frac{1}{\mathcal{W}^2} \odot \nabla R. \tag{3.14}$$

In the rest of this section, we analyze the generalization error using the approximated form of f_R^h as

$$\bar{f}_R^h = \mathbf{I} - \frac{1}{h^2} \odot \nabla R. \tag{3.15}$$

It is easy to verify that $\bar{f}_R^h$ is $\bar{L}_H$-Lipschitz with respect to h. By utilizing Eq. (3.15), we firstly present a lemma to estimate $|\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h)|$.

Lemma 3.8. *Suppose that the mapping $\bar{f}_R^h$ takes the form of $\mathbf{I} - \frac{1}{h^2} \odot \nabla R$ and ℓ_{up} is MSE loss. For any $h \in \mathcal{H}$, we have*

$$|\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h)| \leq \mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}} \left\{ \frac{4B_d}{\sqrt{M}\varepsilon} A_1 + \frac{1}{M\varepsilon^2} A_2 \right\},$$

where

$$A_1 = \left\| \left(\frac{1}{T} \sum_{t=1}^{T} \nabla R_t - \frac{1}{S} \sum_{t=T+1}^{T+S} \nabla R_t \right) (\mathcal{Y}) \right\|, \tag{3.16}$$

$$A_2 = \left| \frac{1}{T} \sum_{t=1}^{T} \|(\nabla R_t)(\mathcal{Y})\|_2^2 - \frac{1}{S} \sum_{t=T+1}^{T+S} \|(\nabla R_t)(\mathcal{Y})\|_2^2 \right|. \tag{3.17}$$

Lemma 3.8 shows that the divergence $|\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h)|$ is bounded by the regularization gradient. The first term A_1 of the upper bound compares an average difference of regularization gradients between all source and target tasks, while the second term A_2 compares such average source-target difference in terms of the norms of their regularization gradients.

Based on Lemma 3.8, we can directly estimate the generalization error $\mathbf{E}_g$ as follows.

Theorem 3.9. *Suppose that Assumption 3.2 and Assumption 3.3 hold, $\{R_t\}_{t=0}^T$ are proper closed and convex, $\bar{f}_R^h$ takes the form of $\mathbf{I} - \frac{1}{h^2} \odot \nabla R$, and ℓ_{up} is MSE loss. Then for any $\delta > 0$, with probability at least $1-\delta$, we have*

$$\mathbf{E}_g \leq 6L_n \bar{L}_H \hat{\mathfrak{G}}_S(\mathcal{H}) + \frac{6B_l}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}} + \mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left\{\frac{8B_d}{\sqrt{M}\varepsilon}A_1 + \frac{2}{M\varepsilon^2}A_2\right\}, \tag{3.18}$$

where A_1 takes the form of (3.16) and A_2 of (3.17).

The upper bound in (3.18) mainly contains two parts. The first leading two terms represent the aforementioned training error upper bound in Theorem 3.7, which constrains the distance between $\hat{h}_{tr}$ and h_{tr}^* and reveals how well $\hat{h}_{tr}$ is learned. The last term is the main part to evaluate the generalization error since they measure the diversity between the source and target model. Both A_1 and A_2 contain the regularization gradient[1] and each of them reveals one kind of "model divergence" between source and target models. Specifically, A_1 directly calculates the difference function between the averaged source regularization gradients and the averaged target regularization gradients. A_2 first calculates the norm value of each regularization gradient on $\mathcal{Y}$, and then calculates the difference between averaged source norm values and averaged target norm values. We think that A_1 mainly cares about the total regularizations contained in the source and target models respectively, while A_2 distinguishes those regularizations included in each model in more detail.

Remark 3.10. When the target and source models are the same, the last term reaches its minimum value, i.e., zero, which trivially means that there is no error brought by the models.

In Sec. 4.5.1, we show some connections between Theorem 3.9 and simple experiments. But it is also faithful to say that the real applications are much more complex and include many other factors that may influence the generalization behavior of h_θ, such as training efficiency. Our theorem 3.9 provides a preliminary perspective to understand the model-level generalization error of the proposed DLW.

4. Experimental results.

We then conduct experiments to verify the effectiveness of our proposed DLW scheme for image denoising models. For convenience, the denoising models implemented with DLWnet h_θ are prefixed by "DLW-". The DLWnet h_θ is simply designed as a four-layer convolutional neural network (CNN).

4.1. Setting of source models.

Three basic denoising models are used as source models to train DLWnet. Their regularization term are nuclear norm (NN) [47], spatial total variation norm (TV) [48], and spectral total variation norm (TVS) [48]. The NN can enforce the bands of image to lie in a low dimensional subspace. TV and TVS characterize the smoothness property in the spatial and spectral dimensions, respectively. These regularizations are widely used for image denoising. And they are also relatively easy to optimize. The ADMM (shown in Algorithm 3.1) is used to solve them due to its fast convergence and simple expression. Algorithm details are included in Appendix B.

[1] based on the approximate form of solution to denoising problem (1.2) because the optimization form itself is intractable to analysis.

Table 1: Setting of source models: There are seven possible combinations. Each of them could train an individual DLWnet h_θ.

	corresponding trained DLWnet names						
source models ↓	N	T	TS	N+T	N+TS	T+TS	N+T+TS
DLW-NN	✓			✓	✓		✓
DLW-TV		✓		✓		✓	✓
DLW-TVS			✓		✓	✓	✓

In the training framework (3.3a)-(3.3a), the above three denoising models can generate totally seven different combinations for the lower-level problem (3.3a) to train the DLWnet, as listed in Table 1. The abbreviations "N", "T" and "TS" mean using DLW-NN, DLW-TV and DLW-TVS as single source model, respectively. The abbreviation "N+T" means that DLW-NN and DLW-TV are used to train the DLWnet and thus the number of models T in (3.3a) is two. Consequently, we derive seven different DLWnets under the seven different model combinations.

4.2. Details of training and testing data. To generate the training pairs, the CAVE dataset[2] [44] is used, which contains 32 images of real-world materials and objects. Specifically, we randomly select 20 images to generate the training pairs. The original size of each image is $512 \times 512 \times 31$, and we crop 2500 overlapping patches with a size of $64 \times 64 \times 31$. After rotating and flipping, the total number of training patch pairs is augmented to 20000. Additionally, seven image datasets are used for testing, including the remaining 10 images in the CAVE dataset, 10 images with a size of $512 \times 512 \times 31$ from ICVL dataset [1], Washington DC Mall data[3] with a size of $200 \times 200 \times 152$, PaviaU data[4] with a size of $340 \times 340 \times 70$, Urban data [17] and Indian Pines data. The first 4 testing datasets are used for synthetic image denoising experiments, and the last two datasets are used for real image denoising.

Firstly, we generate five kinds of complex noise for synthetic experiments. Note that only the noise in case 1 is used to generate the paired "clean/noisy" training patches. In the testing stage, the pairs generated by all five types of noise are used. The details of the generation process of noisy images are as follows:

Case 1 (*Gaussian + Impulse*): Each band in image is corrupted with Gaussian noise, and the noise level is randomly selected from [10,70]. The number of selected bands with additional impulse noise are 10, 10, 40, and 20 for CAVE, ICVL, DC, and PaviaU, respectively. The ratio of impulse noise ranges from [0.1,0.5].

Case 2 (*Gaussian + Stripe*): All the settings are the same as in Case 1, except that the stripe noise is substituted for the impulse noise. The ratio of the stripes ranges from [0.05,0.2].

Case 3 (*Gaussian + Deadline*): All the settings are the same as in Case 1, except that the impulse noise is replaced by the deadline noise, the ratio of which ranges from [0.05,0.2].

Case 4 (*Spatial-Spectral Variant Gaussian*): Each band of the image is corrupted with the spatial-spectral variant Gaussian noise. The noise level for each band is randomly generated

[2] https://cave.cs.columbia.edu/repository/Multispectral
[3] https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html
[4] https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes

by the normal distribution. The range of the noise level is between 10 and 70.

Case 5 (*Mixture*): The images are corrupted with all the noise types from Case 1 to Case 4.

Two quantitative measures are used to evaluate the denoising performance, i.e., peak signal-to-noise ratio (PSNR) and structural similarity (SSIM). The ADAM [18] is used to optimize the network parameters. The epoch number is set as 10. The initial learning rate is $1e^{-3}$ and decays by a factor of 0.8 every epoch. The batch size is set as 10. All the experiments are conducted on a PC with Intel Core i7-8700K CPU, and one GeForce RTX 2080 Ti with 11GB memory. It takes 4∼6 hours to train each DLWnet.

4.3. Weight visualization. It is evident that the implementation principle of DLWnet is to extract a shared explicit weighting scheme across different noisy images and diverse source models. This rationally makes it capable of capturing a unified weighting policy for general noisy images with diverse noise types and different source models with diverse regularization terms. This implies that the extracted weight mapping from the trained DLWnet for a testing image should beneficially deliver both its contained noise information and its possessed structural prior knowledge.

To intuitively illustrate this capability of DLWnet, we visualize the predicted weights calculated by the seven different DLWnets on a typical testing image in Fig. 3. Specifically, one band of the "watercolor" image in the CAVE dataset is corrupted with spatially non-i.i.d Gaussian noise, and the top-right area of the image is more heavily corrupted than the remaining areas. From the figure, the aforementioned analysis can be finely validated. Firstly, the heavy noise information located in the top-right area can be recognized by the weight predicted by all trained DLWnets. The weight values in this noisy area are very small, meaning that all DLWnets successfully recognize this polluted area and attempts to suppress its negative influence on the recovery by setting relatively smaller weights on it. Secondly, all generated weights from DLWnet show evident structural shapes consistent with that of the ground-truth clean image. Especially, more important image structures for recovery, e.g., edges and textures, have been more emphasized by DLWnet through specifying larger weights on them. Besides, the focus of image structures of each weight is also different, implying the underlying impact of different source models on the DLWnet.

4.4. Application of DLWnet. In this section, we select three popular image complex noise removal models as the target models for applying the trained DLWnet. These models are LRTV [15], E3DTV [32] and LRTFDFR [49]. Specifically, LRTV [15] is a matrix-based method, which utilizes both spatial smoothness property and spectral low-rank property. E3DTV [32] is a tensor-based method that explores the sparsity of the base matrix of the gradient map in each mode. The data fidelity terms of LRTV and E3DTV are both ℓ_1 norm. LRTFDFR [49] employs the spectral low-rankness, the group sparsity on the gradient map of the base matrix, and the column continuity of the spectral factor matrix. The data fidelity term of LRTFDFR includes both weighted ℓ_1 norm and ℓ_2 norm.

For each of the three target models, we modify it by simply replacing the data fidelity term with $\|h_\theta(\mathcal{Y})\odot(\mathcal{Y}-\mathcal{X})\|^2$. The modified models implemented with our DLWnet are called DLW-LRTV, DLW-E3DTV and DLW-LRTFDFR, respectively. They are also called *our* methods for convenience. In this section, the type-'N+T+TS' DLWnet is applied for the three DLW-

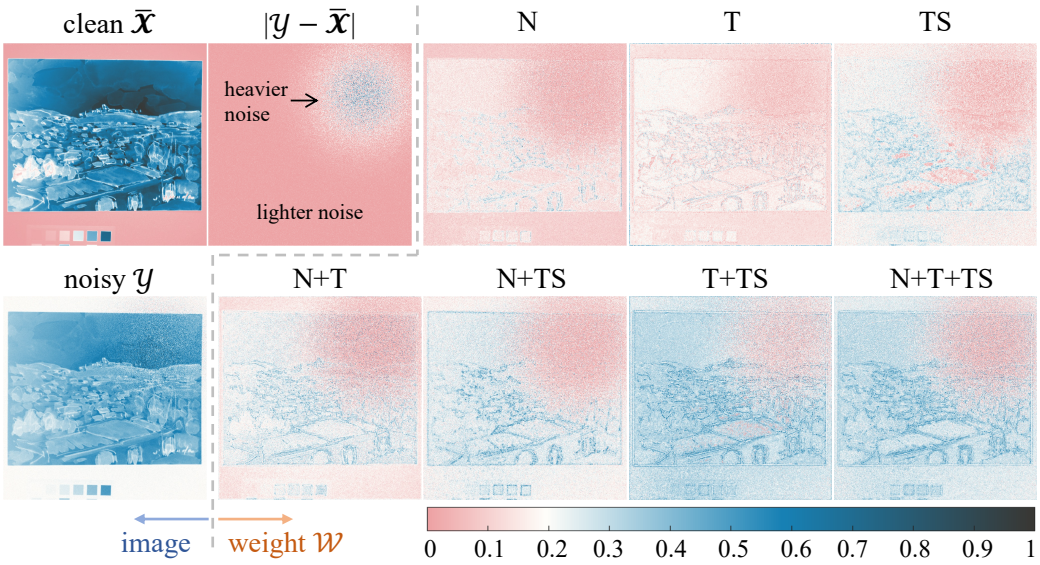


Figure 3: Visual comparison of weight $\mathcal{W}$s predicted by different types of DLWnet. There are two observations: 1) All the seven DLWnets can recognize the heavier noise and assign smaller weights on this area, exactly showing the help-to-denoise function of DLWnet. 2) DLWnets additionally extract some image structures and the structures are different by different DLWnets.

models. Algorithms to solve the DLW-models are inherited from the original model. More specifically DLW-LRTV and DLW-E3DTV are optimized by the ADMM algorithm which is also adopted in the original LRTV and E3DTV models. As for DLW-LRTFDFR, we also adopt the ADMM algorithm because the proximal alternating minimization algorithm used by LRTFDFR can hardly be directly inherited. More details are presented in the Appendix C.

We compare the three DLW-methods with several advanced image complex noise removal methods including LRMR [47], LRTV [15], NMoG [11], HyRes [33], FastHyMix [50], CTV-RPCA [31], E3DTV [32], LRTFDFR [49], HSI-DeNet [9] and image-CNN [45]. The quantitative results are reported in Table 2. From the table, we can easily observe that our DLWnet can dramatically improve the denoising results by comparing LRTV with DLW-LRTV or by comparing E3DTV with DLW-E3DTV. Additionally, it can also be seen that our methods generally perform the best for different types of noise patterns even though only the noise in Case 1 is used in the training process. This verifies that the DLWnet can learn a general weight prediction rule for a wide range of complex noise types and thus can help the model to finely adapt to more noise types. Although FastHyMix achieves the best results in Case 2 (Gaussian+stripe), it does not perform well on the more complex noise in Case 5. We also compare our methods with all competing methods on one real noisy image dataset, Urban. The visual comparison is presented in Fig. 4. It can be observed that the restored image by our DLW-E3DTV method can achieve the best visual result, while other competing methods do not fully remove the noise and the restored image always contains obvious blur effects or

Table 2: Average test performance of different denoising competing methods on the ICVL dataset. The best results in each **column** are in **bold**, and the second best results in each **column** are with underline.

noise →	Case 1	Case 2	Case 3	Case 4	Case 5
index →	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
LRMR	24.35/0.7214	27.77/0.8198	26.45/0.7962	26.15/0.7794	23/0.6868
LRTV	31.67/0.9059	32.91/0.9252	31.34/0.9135	35.49/0.9543	30.3/0.8971
NMoG	28.9/0.8718	30.32/0.9088	29.09/0.9013	26.68/0.8422	24.7/0.7553
HyRes	30/0.8829	35.09/0.9566	32.16/0.9308	32.25/0.9241	26.84/0.8357
FastHyMix	31.09/0.8914	**37.30**/**0.9680**	34.19/**0.9571**	36.23/0.961	27.9/0.8498
CTV-RPCA	31.09/0.8636	30.99/0.8579	30.37/0.8484	28.94/0.7944	28.45/0.7923
E3DTV	34.61/0.9511	34.27/0.9473	33.52/0.9425	32.88/0.9271	31.63/0.9154
LRTFDFR	29.69/0.8376	30.83/0.8479	27.92/0.7729	31.52/0.8553	28.42/0.8213
HSI-DeNet	29.33/0.8588	28.95/0.8389	28.39/0.8308	30.31/0.8843	28.37/0.842
HSI-CNN	34.36/0.932	35.65/0.9551	33.64/0.9456	**37.38**/0.9649	32.02/0.9206
DLW-LRTV	34.94/0.9495	35.62/0.9593	34.52/0.9523	37.00/**0.9692**	**33.90**/**0.9480**
DLW-E3DTV	**35.41**/**0.9529**	35.46/0.959	**35.20**/0.9568	34.55/0.9476	33.26/0.9346
DLW-LRTFDFR	34.09/0.9505	34.44/0.954	33.75/0.9464	35.42/0.964	33.43/0.9469

stripe noise. More experimental results are presented in Appendix C.

From the above synthetic and real image denoising experiments, the effectiveness of our DLWnet is reflected in two aspects. First, the trained DLWnet helps boost the denoising performance of traditional image denoising models. It should be also noted that the DLWnet is not limited to being used on the LRTV, E3DTV and LRTFDFR models, but could be plugged into general weighted image denoising models. Second, the DLWnet inclines to learn a general weight prediction rule to help a model finely fit a wider range of noise types than those involved in DLWnet training.

4.5. Model generalization analysis. In this section, we take a deeper step to explore how DLWnet can behave between various source and target models.

4.5.1. Simple target models. In this experiment, we choose the target models exactly from the source models (i.e. DLW-NN, DLW-TV, and DLW-TVS). Since the divergence between source and target models is relatively clear, we can take a closer look at the generalization ability of all seven DLWnets across different regularizations.

The average experimental results on the ICVL dataset are shown in Table 3. The first row indexes seven types of DLWnets, and the first column indexes the different noise types. For example, the "T" column means that the DLWnet applied for the three target models is trained only using source model DLW-TV. The quantitative results under five noise types are reported.

From Table 3, we can first observe that although the DLWnet is trained only using the noise in Case 1, it can be finely generalized to other types of noises (Cases 2-5), which have not been seen during training. Besides, it seems that some DLWnets always perform better on a specific target model. To explore the underlying rules, we compute the average PSNR value of all the five noise cases based on Table 3, and demonstrate the results in Table 4. Combining the two tables, we have the following observations. First, the corresponding DLWnet inclines

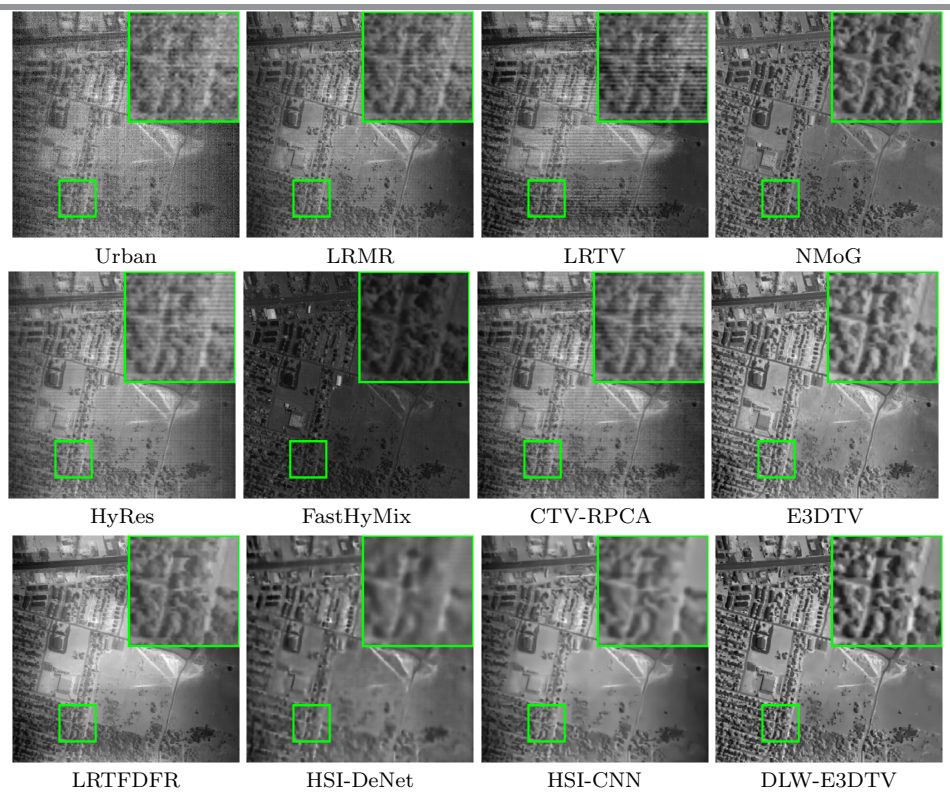


Figure 4: Denoising results by different compared methods on the real noisy image Urban.

to perform the best when the source models and the target models match. For example, in Table 3, if the source model is 'N', the best denoising performance is obtained when the target model is DLW-NN (see second column). The phenomena can also be seen from Table 4 whose diagonal values are the best. Second, the type-'T' DLWnet does not generalize well to the other two models (i.e., 'N' and 'TS'), while type-'N' DLWnet generalizes better than the other two types of DLWnet. Third, although the inconsistency between source and target models degrades the performance, the performance on the target models is still good enough. Fourth, when the number of target models grow, the performance also raises with the number of source models growing (see the last column of Table 4). This is because the DLWnet can capture more regularization information from multiple source models.

We can also get more clues of Theorem 3.9 from Table 4. For convenience, we assume that the training error (first two terms of the upper bound) can be ignored. Denote the sum of the last two terms of the upper bound as

$$U(x,y) := \mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left\{\frac{8B_d}{\sqrt{M}\varepsilon}A_1\right\} + \mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left\{\frac{2}{M\varepsilon^2}A_2\right\},$$
$$\text{where } x = \{\text{source models}\}, y = \{\text{target models}\}. \tag{4.1}$$

Then, a smaller $U(x,y)$ implies that the generalization error is more likely to be small.

Table 3: Average test performance of DLW-NN, DLW-TV and DLW-TVS on ICVL dataset. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

source / noise	N	T	TS	N+T	N+TS	T+TS	N+T+TS
	evaluation index: PSNR/SSIM						
target model 1 : **DLW-NN**							
Case 1	33.2/**0.945**	22.4/0.652	27.4/0.835	32.5/0.941	**33.3**/0.944	26.9/0.829	32.9/0.943
Case 2	33.6/0.949	22.3/0.645	27.8/0.842	33.0/0.943	**33.7**/**0.950**	26.8/0.826	33.4/0.948
Case 3	32.5/**0.946**	22.4/0.650	27.0/0.832	31.8/0.935	**32.6**/0.943	26.5/0.823	32.2/0.941
Case 4	**32.4**/**0.927**	19.0/0.533	25.1/0.759	31.3/0.914	32.1/0.925	23.7/0.732	31.5/0.922
Case 5	**30.4**/**0.903**	19.9/0.549	23.9/0.713	29.5/0.891	29.9/0.895	23.9/0.729	29.6/0.894
target model 2 : **DLW-TV**							
Case 1	29.3/0.836	**32.8**/**0.927**	27.5/0.787	31.5/0.889	29.1/0.834	31.5/0.890	31.0/0.876
Case 2	29.4/0.846	**33.0**/**0.936**	27.7/0.794	31.8/0.899	29.2/0.846	31.7/0.897	31.2/0.886
Case 3	29.0/0.839	**31.9**/**0.928**	26.8/0.776	31.0/0.892	28.8/0.837	30.8/0.888	30.6/0.879
Case 4	28.7/0.827	**31.4**/**0.909**	26.7/0.770	30.9/0.881	28.5/0.828	30.8/0.880	30.4/0.870
Case 5	27.6/0.800	**29.7**/**0.877**	25.2/0.723	29.2/0.856	27.5/0.799	29.0/0.850	29.0/0.843
target model 3 : **DLW-TVS**							
Case 1	31.3/0.913	24.2/0.699	**33.2**/**0.925**	30.5/0.909	32.4/0.924	32.0/0.924	31.8/0.924
Case 2	31.0/0.914	24.0/0.686	**33.5**/**0.929**	30.2/0.907	32.3/0.927	32.1/0.926	31.7/0.925
Case 3	31.3/0.915	24.1/0.694	**32.9**/0.924	30.3/0.906	32.5/**0.926**	31.6/0.919	31.7/0.924
Case 4	29.5/0.883	21.3/0.593	**32.0**/**0.904**	28.7/0.873	31.0/0.902	30.5/0.896	30.28/0.898
Case 5	29.6/0.871	21.9/0.604	30.5/0.878	28.6/0.860	**30.6**/**0.886**	29.5/0.872	29.8/0.881

Table 4: Average PSNR/SSIM values of all five kinds of noise patterns obtained by DLW-NN, DLW-TV and DLW-TVS on ICVL dataset. The best results in each **column** are in **bold**, and the second best results in each **column** are with underline.

target → source ↓	DLW-NN	DLW-TV	DLW-TVS	DLW-NN & DLW-TV	DLW-NN & DLW-TVS	DLW-TV & DLW-TVS	DLW-NN & DLW-TV & DLW-TVS
N	**32.4**/**0.934**	28.8/0.830	30.5/0.899	30.6/0.882	31.5/0.917	29.7/0.864	30.6/0.888
T	21.2/0.606	**31.8**/**0.915**	23.1/0.655	26.5/0.761	22.1/0.631	27.4/0.785	25.3/0.726
TS	26.2/0.796	26.8/0.770	**32.4**/0.912	26.5/0.783	29.3/0.854	29.6/0.841	28.5/0.826
N+T	31.6/0.925	30.9/0.883	29.7/0.891	**31.2**/**0.904**	30.6/0.908	30.3/0.887	30.7/0.900
N+TS	32.3/0.932	28.6/0.829	31.8/**0.913**	30.5/0.880	**32.0**/**0.922**	30.2/0.871	30.9/0.891
T+TS	25.6/0.788	30.8/0.881	31.1/0.907	28.2/0.834	28.4/0.848	**31.0**/**0.894**	29.2/0.859
N+T+TS	31.9/0.930	30.4/0.871	31.1/0.910	31.2/0.900	31.5/0.920	30.8/0.891	**31.1**/**0.904**

When source and target models are the same, we have $U(y, y) = 0$. Correspondingly, in Table 4, the diagonal results are generally the best since the source and target tasks exactly match. When the source and target models are different, there are also some relative observations. Let us first focus on the second column of Table 4. The target model is DLW-NN. By the definition of U, we have $U(\text{N+T, DLW-NN}) = \frac{1}{2}U(\text{T, DLW-NN})$ and $U(\text{N+TS, DLW-NN}) = \frac{1}{2}U(\text{TS, DLW-NN})$. In Table 4, it also shows that the performance of 'N+T' is better than 'T' and 'N+TS' is better than 'TS'. Besides, the performance of 'TS'

is better than 'T', by which we may assume that U(TS, DLW-NN) $<$ U(T, DLW-NN) and thus U(N+TS, DLW-NN) $<$ U(N+T, DLW-NN). Correspondingly, Table 4 shows that the performance of 'N+TS' is better than 'N+T'. More similar observations can be found for other source and target combinations in Table 4. It should be indicated that besides the intrinsics revealed by our learning theory, which inspires to understand the insightful working mechanism underlying our method, especially for its generalization ability, some other practical factors could also affect the generalization behaviour, such as training effectiveness. This is similar to the roles of the conventional learning theory results.

4.5.2. Complex target models. In this experiment, we apply the seven trained DLWnets to LRTV [15], E3DTV [32], and LRTFDFR [49]. The denoising results on the ICVL dataset are illustrated in Table 5. The second column of each sub-table (named "**Original*") in Table 5 records the performance of the original LRTV, E3DTV, and LRTFDFR models. Additionally, we also have the following observations from Table 5. Firstly, most models outperform their original models. This should be attributed to the powerful complex noise description ability of the data fidelity term $\|h_\theta(\mathcal{Y})\odot(\mathcal{Y}-\mathcal{X})\|^2$. Secondly, although only "Gaussian+impulse" noise is used for training the DLWnet, the DLW-models whose weight is predicted by the trained DLWnet can also achieve superior results than the original models in other noise cases, which further proves the effectiveness of our DLWnet to handle complex noise. This can be rationally explained by that our DLWnet has learned the essential noise characteristics from the training data and thus can be well generalized to other noise types.

Table 6 averages the PSNR/SSIM results of all five noise cases. From Tables 5 and 6, we can observe that for DLW-LRTV, the type-'N+T+TS' and type-'N+T' DLWnets obtain the best and second best performance, which is reasonable since LRTV mainly considers the low-rankness and spatial smoothness property, complying with the 'N' regularization and 'T' regularization, respectively. As for DLW-E3DTV and DLW-LRTFDFR, the last two DLWnets achieve the best results, which should be attributed to the fact that the DLWnet learned from diverse models can generally capture more image priors, and thus can generalize well to a wide range of new target models. Additionally, we also notice that the type-'T' DLWnet gains the lowest performance among the original and other models. The reason might be that the TV regularization only grabs the spatial features while neglecting the spectral correlation property, which has been verified to be of great importance for image denoising.

Further, Figure 5 shows the visual comparison of LRTV and its corresponding models with different DLWnets. It can be easily seen that except for type-'T' DLWnet, the proposed models with other DLWnets can effectively remove the complex noise.

We conducted a classification experiment on the Indian Pines dataset. The scene in this dataset contains 16 classes and the sample number of each class ranges from 20 to 2455. The classifier for this experiment is the basic support vector machine (SVM). For each class, we randomly select 15 samples for training and the remaining samples are used for testing. For each method, the experiment is conducted 10 times. The best result for each method is presented in Figure 6. The overall accuracy is used to measure the classification results. From Figure 6, we can see that the denoised image of the model with the proposed weighting scheme achieves higher classification accuracy than that of its corresponding model, indicating that the proposed automatic weighting scheme can help improve the denoising performance.

Table 5: Complex target models: average test performance of LRTV, E3DTV and LRTFDFR and their corresponding DLW-models on ICVL dataset. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

source / noise	**Original*	N	T	TS	N+T	N+TS	T+TS	N+T +TS
	evaluation index: PSNR/SSIM							
target model 1 : **LRTV**								
Case 1	31.67/ 0.9059	34.44/ 0.9419	28.76/ 0.8152	31.45/ 0.9122	34.54/ **0.9503**	34.30/ 0.9417	34.31/ 0.9416	**34.94**/ 0.9495
Case 2	32.91/ 0.9252	35.23/ 0.9538	29.23/ 0.8353	32.73/ 0.9275	35.41/ 0.9592	35.06/ 0.9542	35.05/ 0.9496	**35.62**/ **0.9593**
Case 3	31.34/ 0.9135	34.21/ 0.9476	28.85/ 0.8301	31.01/ 0.9179	34.22/ **0.9531**	34.08/ 0.9462	33.57/ 0.9435	**34.52**/ 0.9523
Case 4	35.49/ 0.9543	36.61/ 0.9645	32.25/ 0.9145	34.44/ 0.9520	36.90/ **0.9709**	36.59/ 0.9658	36.61/ 0.9686	**37.00**/ 0.9692
Case 5	30.30/ 0.8971	33.57/ 0.9418	28.94/ 0.8346	30.30/ 0.9102	33.34/ 0.9467	33.55/ 0.9426	33.07/ 0.9432	**33.90**/ **0.9480**
target model 2 : **E3DTV**								
Case 1	34.61/ 0.9511	34.97/ 0.9474	30.34/ 0.8741	34.37/ 0.9396	35.16/ **0.9563**	34.95/ 0.9467	**35.70**/ 0.9551	35.41/ 0.9529
Case 2	34.27/ 0.9473	35.04/ 0.9532	29.76/ 0.8571	34.77/ 0.9474	35.18/ 0.9605	35.10/ 0.9542	**35.81**/ **0.9613**	35.46/ 0.9590
Case 3	33.52/ 0.9425	35.06/ 0.9540	29.59/ 0.8591	33.53/ 0.9409	34.63/ **0.9570**	35.10/ 0.9529	34.83/ 0.9560	**35.20**/ 0.9568
Case 4	32.88/ 0.9271	34.12/ 0.9420	27.81/ 0.7918	33.53/ 0.9331	34.13/ 0.9478	34.19/ 0.9425	**34.77**/ **0.9492**	34.55/ 0.9476
Case 5	31.63/ 0.9154	33.21/ 0.9307	27.75/ 0.7888	31.15/ 0.9110	32.61/ 0.9337	33.23/ 0.9300	32.77/ 0.9318	**33.26**/ **0.9346**
target model 3 : **LRTFDFR**								
Case 1	29.69/ 0.8376	33.63/ 0.9421	27.97/ 0.8116	32.03/ 0.9224	33.70/ 0.9506	33.81/ 0.9444	**34.16**/ **0.9526**	34.09/ 0.9505
Case 2	30.83/ 0.8479	34.21/ 0.9496	28.66/ 0.8374	32.82/ 0.9287	34.17/ 0.9523	34.03/ 0.9491	**34.56**/ 0.9525	34.44/ **0.9540**
Case 3	27.92/ 0.7729	33.25/ 0.9402	28.29/ 0.8267	31.25/ 0.9064	33.10/ 0.9418	33.37/ 0.9406	33.43/ 0.9426	**33.75**/ **0.9464**
Case 4	31.52/ 0.8553	35.41/ 0.9617	31.12/ 0.9096	34.70/ 0.9534	35.23/ **0.9642**	35.23/ 0.9614	**35.51**/ 0.9630	35.42/ 0.9640
Case 5	28.42/ 0.8213	33.00/ 0.9382	28.41/ 0.8340	31.61/ 0.9248	32.70/ 0.9436	33.22/ 0.9413	33.22/ **0.9475**	**33.43**/ 0.9469

4.5.3. Special target models: deep image prior. As mentioned in subsection 3.3, deep image prior problem of form (3.7) can be seen as a special kind of target models. In this experiment, we use S2DIP [23] as the backbone model since it is specifically designed for image denoising.

Table 7 shows the experimental results on the CAVE dataset in five noise cases. Each restoration experiment is repeated 3 times and the average results are recorded. In Table 7, the "max" column presents the maximum output PSNR value during training. The "final" column means the average PSNR values of the last 100 iterations, and also represents the converged results. "|d.v|" is the absolute difference value between the maximum and the final values. From Table 7, we can see that the DLW-S2DIP model outperforms the original model

Table 6: Average PSNR/SSIM values of all five kinds of noise patterns obtained by LRTV, E3DTV, LRTFDFR and their corresponding DLW-models on ICVL dataset.

source \ target	LRTV	E3DTV	LRTFDFR
**Original*	32.34/0.9192	33.38/0.9367	29.68/0.827
N	34.81/0.9499	34.48/0.9455	33.9/0.9464
T	29.61/0.8459	29.05/0.8342	28.89/0.8439
TS	31.99/0.924	33.47/0.9344	32.48/0.9271
N+T	34.88/**0.9560**	34.34/**0.9511**	33.78/0.9505
N+TS	34.72/0.9501	34.51/0.9453	33.93/0.9474
T+TS	34.52/0.9493	**34.78**/0.9507	34.18/0.9516
N+T+TS	**35.20**/0.9557	34.78/0.9502	**34.23**/**0.9524**

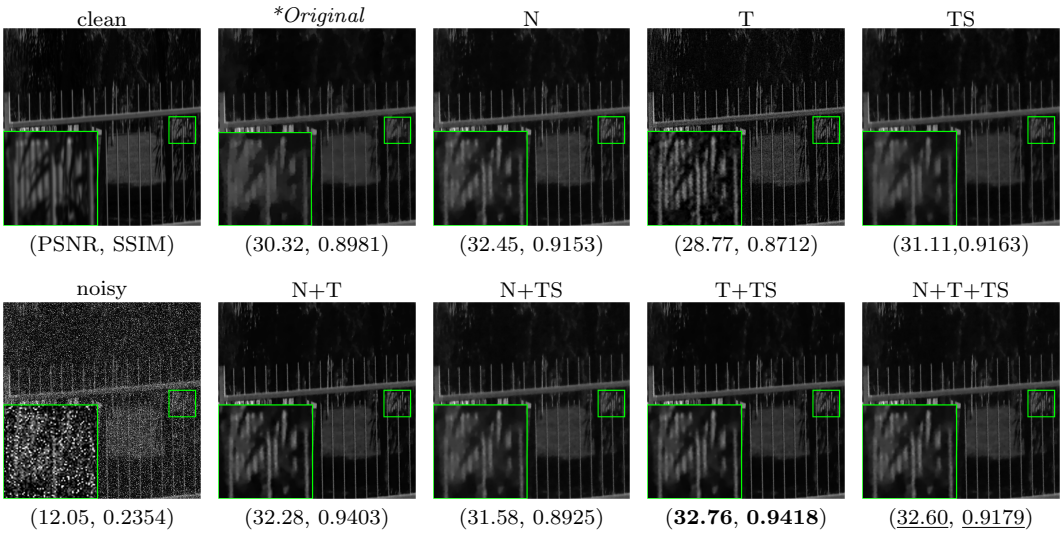


Figure 5: Denoising results of LRTV and its corresponding DLW-models on ICVL dataset. The noisy type is "Gaussian+impulse"

in most noise cases. For example, most of the DLW-S2DIP models perform better than the original model in the complex noise case (i.e., Case 5). Besides, the "|d.v|" value of the DLW-S2DIP model is always smaller than S2DIP, which reveals that the DLW-S2DIP models are more stable than the original model. A different observation from previous experiments is that the type-'T' DLWnet doesn't perform the worst among all DLWnets. This is perhaps due to the extra implicit image prior brought by network g_η. To illustrate the convergence, we also plot the PSNR tendency during the training process in fig-dip-converge. Except for type-'TS' DLWnet, DLW-S2DIP with other DLWnets converges very quickly in the early stage. In Case 5, we see that the DLW-S2DIP with type-'N+T+TS' DLWnet suffers from one big vibration at about the 6000_{th} iteration, which causes a large "|d.v|" value 0.424 in Table 7. However, its performance keeps rising after the 6000_{th} iteration and finally reaches the third best.

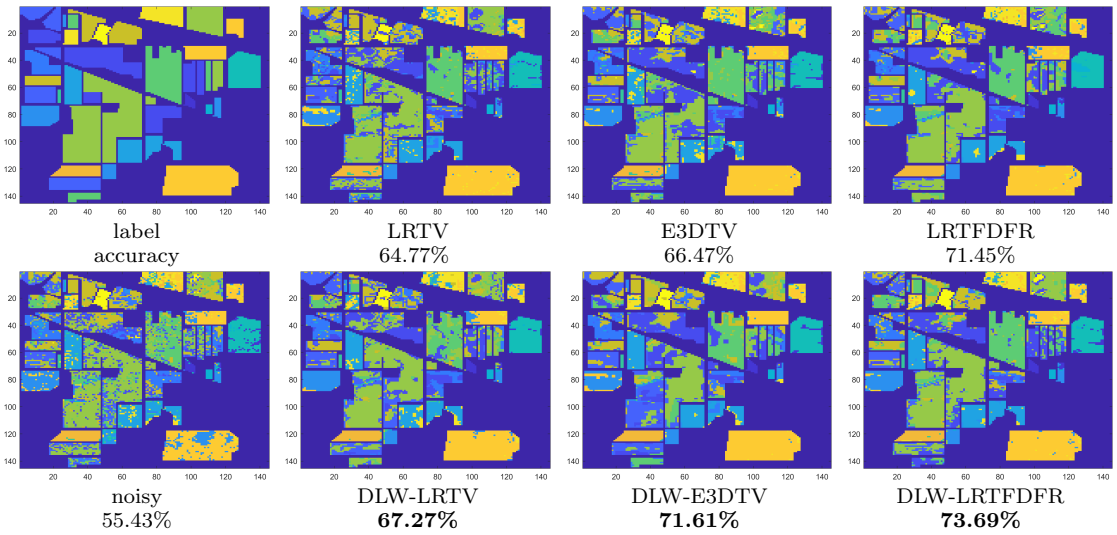


Figure 6: Classification results of the compared methods on the Indian Pines dataset.

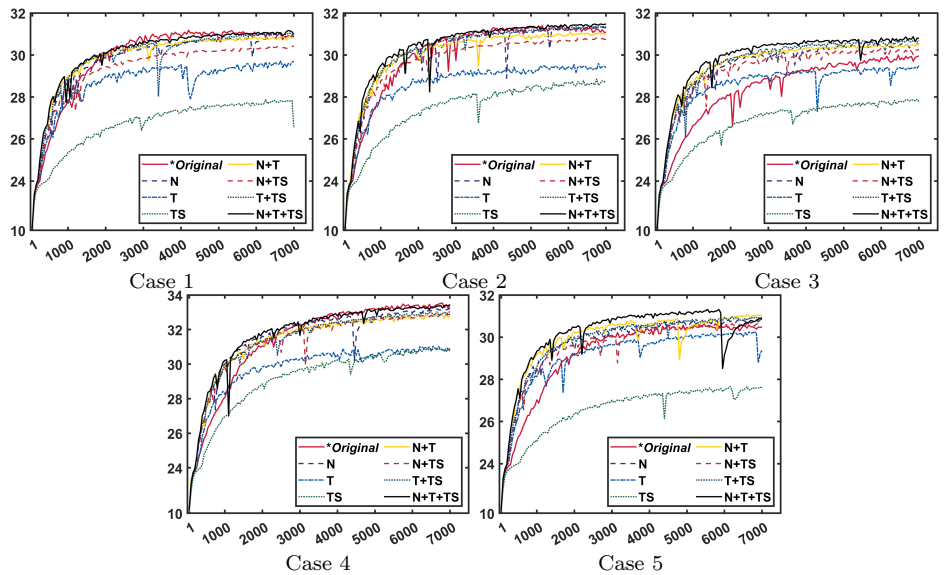


Figure 7: The tendency of PSNR values during iterations for S2DIP and DLW-S2DIP with different DLWnets.

5. Conclusions. In this work, we have proposed an automatic weighting scheme for the weighted image denoising models. The weight is predicted by an explicit neural network mapping called DLWnet, which can be learned under a bi-level optimization framework. The trained DLWnet has been validated to be able to generalize to other weighted image denoising models in a plug-and-play manner. Experimental results have substantiated that the proposed

Table 7: PSNR results of "jelly_beans" in CAVE dataset for S2DIP and DLW-S2DIP with different DLWnets. The best results in each **column** are in **bold**, and the second best results in each **column** are with underline.

		target model: S2DIP							
source / noise		**Original*	N	T	TS	N+T	N+TS	T+TS	N+T +TS
Case 1	max	**31.20**	30.95	29.85	27.94	30.93	30.46	31.1	31.16
	final	30.91	30.87	29.62	27.48	30.84	30.37	31.02	**31.09**
	|d.v|	0.282	**0.074**	0.232	0.461	0.088	0.095	0.078	0.076
Case 2	max	31.41	31.35	29.63	28.9	31.11	30.85	31.43	**31.54**
	final	31.18	31.28	29.41	28.72	31.02	30.77	31.34	**31.45**
	|d.v|	0.229	**0.076**	0.226	0.176	0.087	0.083	0.09	0.081
Case 3	max	30.13	30.6	29.5	27.96	30.58	30.35	30.80	**30.87**
	final	29.89	30.47	29.37	27.83	30.45	30.22	30.67	**30.78**
	|d.v|	0.231	0.137	0.125	0.139	0.123	0.13	0.131	**0.097**
Case 4	max	**33.60**	33.26	31.18	31.05	32.99	32.94	33.1	33.48
	final	**33.44**	33.17	30.89	30.92	32.83	32.86	33	33.37
	|d.v|	0.155	0.092	0.29	0.13	0.165	**0.085**	0.096	0.117
Case 5	max	30.67	30.94	30.31	27.72	31.12	30.74	31.02	**31.33**
	final	30.43	30.8	29.48	27.62	**31.03**	30.62	30.9	30.91
	|d.v|	0.242	0.145	0.831	0.100	**0.092**	0.125	0.113	0.424

DLWnet can help a new weighted image denoising model to finely adapt to complex noise types and thus can generalize well to different image datasets. Also, experimental results validated that the DLWnet trained with diverse regularizers (i.e., models) can be used for new models to further improve their denoising performance. Additionally, we have further theoretically proved a generalization error upper bound of the DLWnet when it is plugged into a new weighted denoising model, showing its insightfully possessed generalization capability.

The proposed automatic weighting scheme still has several limitations. Firstly, the learning of the DLWnet requires the iterative optimization algorithm of solving the lower-level problem to be differentiable w.r.t. weight $\mathcal{W}$. However, some common operators only have sub-gradients, such as nuclear norm and total variation (TV) norm, which may make the training of DLWnet unstable. Secondly, the theoretical analysis of generalization error is based on an approximation to the solution to the source models. Besides, the source and target models are symmetric in $U(x, y)$, i.e., $U(x, y) = U(y, x)$. However, in practice it could be different when exchanging the source and target tasks.

Additionally, how to further improve the generalization ability of DLWnet still needs to be more deeply and comprehensively investigated. Some possible strategies may help, such as constraining specific structures on the predicted parameters [35]. Furthermore, we will consider ways to address and incorporate more physical insights underlying images noise to further improve the quality of our extracted weighting scheme and enhance its denoising effect. We will investigate these issues in our future research.

Appendix A. Proof of Lemmas and Theories.

Proof of Lemma 3.1. Let $\ell(\mathcal{X}) := \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{X})\|^2$. For any $\eta \in [0, 1]$, we have that

$$\begin{aligned}
&\eta\ell(\mathcal{X}) + (1-\eta)\ell(\mathcal{Z}) - \ell(\eta\mathcal{X} + (1-\eta)\mathcal{Z}) \\
=&\frac{\eta}{2}\|\mathcal{W} \odot \mathcal{X}\|^2 + \frac{1-\eta}{2}\|\mathcal{W} \odot \mathcal{Z}\|^2 - \frac{1}{2}\|\mathcal{W} \odot (\eta\mathcal{X} + (1-\eta)\mathcal{Z})\|^2 \\
=&\frac{1}{2}\eta(1-\eta)\|\mathcal{W} \odot \mathcal{X}\|^2 + \frac{1}{2}\eta(1-\eta)\|\mathcal{W} \odot \mathcal{Z}\|^2 - \eta(1-\eta)\langle \mathcal{W} \odot \mathcal{X}, \mathcal{W} \odot \mathcal{Z}\rangle \\
=&\frac{1}{2}\eta(1-\eta)\|\mathcal{W} \odot (\mathcal{X} - \mathcal{Z})\|^2 \\
\geq&\frac{\varepsilon^2}{2}\eta(1-\eta)\|\mathcal{X} - \mathcal{Z}\|^2.
\end{aligned}$$

Thus, the objective function of (1.2) is a closed and strongly convex function as a sum of the closed and ε^2-strongly convex function $\ell(\mathcal{X})$ and the closed and convex function $R(\mathcal{X})$. Similar to Theorem 6.3 in [3], Thus the solution to (1.2) is unique. ∎

Proof of Lemma 3.5. Let $\mathcal{X}_1 = \hat{\mathcal{X}}(\mathcal{Y}, \mathcal{W}_1, R)$ and $\mathcal{X}_2 = \hat{\mathcal{X}}(\mathcal{Y}, \mathcal{W}_2, R)$ for short. By definition,

$$\begin{aligned}
\mathcal{X}_1 &= \arg\min_{\mathcal{X}} \ \frac{1}{2}\|\mathcal{W}_1 \odot (\mathcal{Y} - \mathcal{X})\|^2 + R(\mathcal{X}), \\
\mathcal{X}_2 &= \arg\min_{\mathcal{X}} \ \frac{1}{2}\|\mathcal{W}_2 \odot (\mathcal{Y} - \mathcal{X})\|^2 + R(\mathcal{X}).
\end{aligned}$$

Applying Fermat's optimality condition, we have that there exists $\mathcal{G}_1 \in \partial R(\mathcal{X}_1)$ and $\mathcal{G}_2 \in \partial R(\mathcal{X}_2)$, such that

$$\begin{aligned}
\mathcal{W}_1^2 \odot (\mathcal{X}_1 - \mathcal{Y}) + \mathcal{G}_1 = 0, \\
\mathcal{W}^2 \odot (\mathcal{X}_2 - \mathcal{Y}) + \mathcal{G}_2 = 0.
\end{aligned}$$

By the definition of subgradient, we also have

$$\begin{aligned}
R(\mathcal{X}_2) - R(\mathcal{X}_1) \geq&\langle \mathcal{G}_1, \mathcal{X}_2 - \mathcal{X}_1\rangle, \\
R(\mathcal{X}_1) - R(\mathcal{X}_2) \geq&\langle \mathcal{G}_2, \mathcal{X}_1 - \mathcal{X}_2\rangle.
\end{aligned}$$

Combining the above two equations, we get

$$\begin{aligned}
0 \geq& \langle \mathcal{G}_2 - \mathcal{G}_1, \mathcal{X}_1 - \mathcal{X}_2\rangle, \\
=& \left\langle \mathcal{W}^2 \odot (\mathcal{Y} - \mathcal{X}_2) - \mathcal{W}_1^2 \odot (\mathcal{Y} - \mathcal{X}_1), \mathcal{X}_1 - \mathcal{X}_2 \right\rangle,
\end{aligned}$$

which gives

$$\begin{aligned}
\left\langle \mathcal{Y} \odot (\mathcal{W}_1^2 - \mathcal{W}^2), \mathcal{X}_1 - \mathcal{X}_2 \right\rangle \geq& \left\langle \mathcal{W}_1^2 \odot \mathcal{X}_1 - \mathcal{W}^2 \odot \mathcal{X}_2, \mathcal{X}_1 - \mathcal{X}_2 \right\rangle, \\
=& \langle \mathcal{W}_1^2 \odot (\mathcal{X}_1 - \mathcal{X}_2) + (\mathcal{W}_1^2 - \mathcal{W}^2) \odot \mathcal{X}_2, \mathcal{X}_1 - \mathcal{X}_2\rangle, \\
\Rightarrow \langle (\mathcal{Y} - \mathcal{X}_2) \odot (\mathcal{W}_1^2 - \mathcal{W}^2), &\mathcal{X}_1 - \mathcal{X}_2\rangle \geq \langle \mathcal{W}_1^2 \odot (\mathcal{X}_1 - \mathcal{X}_2), \mathcal{X}_1 - \mathcal{X}_2\rangle.
\end{aligned} \tag{A.1}$$

For the right side of (A.1), we have

$$\langle \mathcal{W}_1^2 \odot (\mathcal{X}_1 - \mathcal{X}_2), \mathcal{X}_1 - \mathcal{X}_2 \rangle \geq \varepsilon^2 \|\mathcal{X}_1 - \mathcal{X}_2\|^2 \geq 0.$$

Thus,

$$\begin{aligned}
\varepsilon^2 \|\mathcal{X}_1 - \mathcal{X}_2\|^2 &\leq \langle (\mathcal{Y} - \mathcal{X}_2) \odot (\mathcal{W}_1^2 - \mathcal{W}^2), \mathcal{X}_1 - \mathcal{X}_2 \rangle \\
&\leq \|(\mathcal{Y} - \mathcal{X}_2) \odot (\mathcal{W}_1^2 - \mathcal{W}^2)\| \|\mathcal{X}_1 - \mathcal{X}_2\| \\
\Rightarrow \varepsilon^2 \|\mathcal{X}_1 - \mathcal{X}_2\| &\leq \|(\mathcal{Y} - \mathcal{X}_2) \odot (\mathcal{W}_1 + \mathcal{W}_2) \odot (\mathcal{W}_1 - \mathcal{W}_2)\| \\
&\leq 4 B_d B_w \|\mathcal{W}_1 - \mathcal{W}_2\|, \\
\Rightarrow \|\mathcal{X}_1 - \mathcal{X}_2\| &\leq \frac{4 B_d B_w}{\varepsilon^2} \|\mathcal{W}_1 - \mathcal{W}_2\|.
\end{aligned}$$

The proof is then completed. ∎

Proof of Theorem 3.7. The proof consists of several parts. We will firstly present an error bound [28][36] and then decouple the complexity of $\mathcal{F}_{\mathcal{H}}$ using Lemma 3.5.

1) Let $\mathcal{Q} := Q_1 \otimes ... \otimes Q_T$ be Cartesian product of T function classes Q_ts, with element $\boldsymbol{q} = (q_1, ..., q_T)$ and $q_t \in Q_t$ for all $t \in [T]$. The function q_t in Q_t satisfies $0 \leq q_t \leq B_l$ for all $t \in [T]$. The data set $\mathbf{S} = \{z_{t,i_t}\}_{i_t=1}^{N_t}{}_{t=1}^{T}$ consists of $\sum_{t=1}^{T} N_t$ data points i.i.d sampled from distribution D. For a $\boldsymbol{q} \in \mathcal{Q}$, let $\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}] = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i_t=1}^{N_t} q_t(z_{t,i_t})$, and $\mathbb{E}[\boldsymbol{q}] = \mathbb{E}[\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}]] = \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{z \sim \mathrm{D}}[q_t(z)]$.

We then define

$$\Phi(\mathbf{S}) := \sup_{\boldsymbol{q} \in \mathcal{Q}} \left(\mathbb{E}[\boldsymbol{q}] - \hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}] \right).$$

Let $\mathbf{S}'$ be another dataset that is identical to $\mathbf{S}$ except for the only element at the (t, i_t)-th position, i.e., $z'_{t,i_t} \neq z_{t,i_t}$ where $z'_{t,i_t} \in \mathbf{S}'$ and $z_{t,i_t} \in \mathbf{S}$. Then we have

$$\begin{aligned}
\Phi(\mathbf{S}') - \Phi(\mathbf{S}) &\leq \sup_{\boldsymbol{q} \in \mathcal{Q}} \left(\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}] - \hat{\mathbb{E}}_{\mathbf{S}'}[\boldsymbol{q}] \right), \\
&= \sup_{\boldsymbol{q} \in \mathcal{Q}} \left(\frac{q_t(z_{t,i_t}) - q_t(z'_{t,i_t})}{T N_t} \right), \\
&\leq \frac{B_l}{T N_t}.
\end{aligned}$$

According to McDiarmid's inequality, we have

$$\mathbb{P}\left[\Phi(\mathbf{S}) - \mathbb{E}_{\mathbf{S}}[\Phi(\mathbf{S})] \geq \epsilon \right] \leq \exp\left(\frac{T^2}{B_l^2} \frac{-2\epsilon^2}{\sum_{t=1}^{T} \frac{1}{N_t}} \right).$$

Let $\frac{\delta}{2} = \exp\left(\frac{T^2}{B_l^2} \frac{-2\epsilon^2}{\sum_{t=1}^{T} \frac{1}{N_t}} \right)$. The above inequality is equal to say that with probability at least $1 - \frac{\delta}{2}$, the following inequality holds:

$$\Phi(\mathbf{S}) \leq \mathbb{E}_{\mathbf{S}}[\Phi(\mathbf{S})] + \frac{B_l}{T} \sqrt{\sum_{t=1}^{T} \frac{1}{N_t}} \sqrt{\frac{1}{2} \log \frac{2}{\delta}}. \tag{A.2}$$

Next, we estimate $\mathbb{E}_{\mathbf{S}}[\Phi(\mathbf{S})]$ as follows:

$$\begin{aligned}
\mathbb{E}_{\mathbf{S}}[\Phi(\mathbf{S})] &= \mathop{\mathbb{E}}_{\mathbf{S}}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\left(\mathbb{E}[\boldsymbol{q}]-\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}]\right)\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{S}}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\mathbb{E}_{\mathbf{S}'}\left(\hat{\mathbb{E}}_{\mathbf{S}'}[\boldsymbol{q}]-\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}]\right)\right] \\
&\leq \mathop{\mathbb{E}}_{\mathbf{S},\mathbf{S}'}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\left(\hat{\mathbb{E}}_{\mathbf{S}'}[\boldsymbol{q}]-\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}]\right)\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{S},\mathbf{S}'}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\left(q_t(z'_{t,i_t})-q_t(z_{t,i_t})\right)\right] \\
&= \mathop{\mathbb{E}}_{\Sigma,\mathbf{S},\mathbf{S}'}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sigma_{t,i_t}\left(q_t(z'_{t,i_t})-q_t(z_{t,i_t})\right)\right] \\
&\leq \mathop{\mathbb{E}}_{\Sigma,\mathbf{S}'}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sigma_{t,i_t}q_t(z'_{t,i_t})\right]+\mathop{\mathbb{E}}_{\Sigma,\mathbf{S}}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}-\sigma_{t,i_t}q_t(z_{t,i_t})\right] \\
&= 2\mathop{\mathbb{E}}_{\Sigma,\mathbf{S}}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sigma_{t,i_t}q_t(z_{t,i_t})\right] \\
&= 2\mathfrak{R}(\mathcal{Q}).
\end{aligned}$$

Checking the conditions of McDiarmid's inequality again to $\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q})$, that is

$$\hat{\mathfrak{R}}'_{\mathbf{S}}(\mathcal{Q})-\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q})\leq\mathop{\mathbb{E}}_{\Sigma}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}}\frac{\sigma_{t,i_t}\left(q(z'_{t,i_t})-q(z_{t,i_t})\right)}{TN_t}\right]\leq\frac{B_l}{TN_t}.$$

Thus, with probability at least $1-\frac{\delta}{2}$, the following inequality holds:

$$\mathfrak{R}(\mathcal{Q})\leq\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q})+\frac{B_l}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{1}{2}\log\frac{2}{\delta}}. \tag{A.3}$$

Combining (A.2) and (A.3), we have that with probability at least $1-\delta$, the following inequality holds:

$$\mathbb{E}[\boldsymbol{q}]\leq\hat{\mathbb{E}}_{\mathbf{S}}[\boldsymbol{q}]+2\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q})+3\frac{B_l}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{1}{2}\log\frac{2}{\delta}}. \tag{A.4}$$

The first part of the proof is then completed. Just let $z_{t,i_t}=(y_{t,i_t},x_{t,i_t})$ and $q_t(z_{t,i_t})=\ell_{up}(f_t^h(y_{t,i_t}),x_{t,i_t})$, we will get very close to the final training error estimation. But let us first decouple $\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q})$ by some so-called contraction inequalities and Lemma 3.5, which leads to the second part of the proof.

2) Let $Q_t = \{q_t(y,x) = \ell_{up}(f_t^h(y),x) | h \in \mathcal{H}, y \in \mathbb{Y}, x \in \mathbb{X}\}$.

$$\begin{aligned}\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q}) &= \mathop{\mathbb{E}}_{\Sigma}\left[\sup_{\boldsymbol{q}\in\mathcal{Q}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sigma_{t,i_t}q_t(z_{t,i_t})\right] \\ &= \mathop{\mathbb{E}}_{\Sigma}\left[\sup_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sigma_{t,i_t}\ell_{up}\left(f_t^h(y_{t,i_t}),x_{t,i_t}\right)\right].\end{aligned}$$

According to [25] and Assumption 3.3, we apply the vector-contraction inequality and get

$$\begin{aligned}&\mathbb{E}_{\Sigma}\left[\sup_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sigma_{t,i_t}\ell_{up}(f_t^h(y_{t,i_t}),x_{t,i_t})\right] \leq \\ &\sqrt{\frac{\pi}{2}}L_n\mathbb{E}_G\left[\sup_{h\in\mathcal{H}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M} g_{ti_tm}f_t^h(y_{t,i_t})_m\right],\end{aligned}$$

where $g_{ti_tm} \overset{i.i.d}{\sim} \mathcal{N}(0,1)$, G is the set of all g_{ti_tm}s and $f_t^h(y_{t,i_t})_m$ is the m-th element of $f_t^h(y_{t,i_t})$. The above inequality implies that

$$\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{G}) \leq \sqrt{\frac{\pi}{2}}L_n\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}}) \leq \frac{3}{2}L_n\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}}).$$

We could further decouple $\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}})$ using Lemma 3.5.
Similar to [2], we define two zero mean Gaussian processes

$$\begin{aligned}X_h &= \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M} g_{ti_tm}f_t^h(y_{t,i_t})_m, \\ Y_h &= \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M} \bar{g}_{ti_tm}L_H h(y_{t,i_t})_m.\end{aligned}$$

Then it can be deduced that:

$$
\begin{aligned}
&\mathbb{E}_G\left[(X_h - X_{h'})^2\right] \\
=&\mathbb{E}_G\left[\left[\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M}\left(f_t^h(y_{t,i_t})_m - f_t^{h'}(y_{t,i_t})_m\right)g_{ti_tm}\right]^2\right] \\
=&\frac{1}{T^2}\sum_{t=1}^{T}\frac{1}{N_t^2}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M}\left(f_t^h(y_{t,i_t})_m - f_t^{h'}(y_{t,i_t})_m\right)^2 \\
=&\frac{1}{T^2}\sum_{t=1}^{T}\frac{1}{N_t^2}\sum_{i_t=1}^{N_t}\|f_t^h(y_{t,i_t}) - f_t^{h'}(y_{t,i_t})\|^2 \\
\leq&\frac{1}{T^2}\sum_{t=1}^{T}\frac{1}{N_t^2}\sum_{i_t=1}^{N_t}L_H^2\|h(y_{t,i_t}) - h'(y_{t,i_t})\|^2 \\
=&\frac{1}{T^2}\sum_{t=1}^{T}\frac{1}{N_t^2}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M}L_H^2\left(h(y_{t,i_t})_m - h'(y_{t,i_t})_m\right)^2 \\
=&\mathbb{E}_{\bar{G}}\left[\left[\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i_t=1}^{N_t}\sum_{m=1}^{M}L_H\left(h(y_{t,i_t})_m - h'(y_{t,i_t})_m\right)\bar{g}_{ti_tm}\right]^2\right] \\
=&\mathbb{E}_{\bar{G}}\left[(Y_h - Y_{h'})^2\right].
\end{aligned}
$$

The inequality above is derived by Lemma 3.5. Note that $\mathcal{H}$ is a function class parameterized by a finite number of real-valued θ. Then applying the Sudakov-Fernique inequality to the general Gaussian process, we get

$$
\mathbb{E}_G\left[\sup_{h\in\mathcal{H}}X_h\right] \leq \mathbb{E}_G\left[\sup_{h\in\mathcal{H}}Y_h\right],
$$

which is equivalent to

$$
\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}}) \leq L_H\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{H}).
$$

The second part of the proof is then completed. Next, we will finally prove the proposed training error bound.

3)

$$
\begin{aligned}
\mathbf{E}_{tr} &= \mathbf{R}_{tr}(\hat{h}_{tr}) - \mathbf{R}_{tr}(h_{tr}^*) \\
&= \underbrace{\mathbf{R}_{tr}(\hat{h}_{tr}) - \hat{\mathbf{R}}_{tr}(\hat{h}_{tr})}_{(a)} + \underbrace{\hat{\mathbf{R}}_{tr}(\hat{h}_{tr}) - \hat{\mathbf{R}}_{tr}(h_{tr}^*)}_{(b)} + \underbrace{\hat{\mathbf{R}}_{tr}(h_{tr}^*) - \mathbf{R}_{tr}(h_{tr}^*)}_{(c)}.
\end{aligned}
$$

According to the definition of $\hat{h}_{tr}$, $(b) \leq 0$. For $(a)+(c)$, we directly estimate it by the result (A.4) in the first part of this proof. With probability at least $1-\delta$, the following inequality

holds

$$
\begin{aligned}
(a)+(c) &\le 4\hat{\mathfrak{R}}_{\mathbf{S}}(\mathcal{Q}) + 6\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{1}{2}\log\frac{\delta}{2}} \\
&\le 6L_n\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{F}_{\mathcal{H}}) + 6\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{1}{2}\log\frac{\delta}{2}} \\
&\le 6L_nL_H\hat{\mathfrak{G}}_{\mathbf{S}}(\mathcal{H}) + 6\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{N_t}}\sqrt{\frac{1}{2}\log\frac{\delta}{2}}.
\end{aligned}
$$

The whole proof is then completed. ∎

Proof of Lemma 3.8. For ease of notation, let $u_t^h = \frac{}{h} \odot \nabla R_t$.

$$
\begin{aligned}
\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h) &= \frac{1}{M}\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left[\frac{1}{S}\sum_{t=T+1}^{T+S}\|\mathcal{Y}-u_t^h(\mathcal{Y})-\mathcal{X}\|_2^2 - \frac{1}{T}\sum_{t=1}^{T}\|\mathcal{Y}-u_t^h(\mathcal{Y})-\mathcal{X}\|_2^2\right] \\
&= \frac{1}{M}\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left[\frac{1}{S}\sum_{t=T+1}^{T+S}\|u_t^h(\mathcal{Y})\|_2^2 - \frac{1}{T}\sum_{t=1}^{T}\|u_t^h(\mathcal{Y})\|_2^2\right. \\
&\quad \left. + 2\left\langle \mathcal{Y}-\mathcal{X}, \frac{1}{T}\sum_{t=1}^{T}u_t^h(\mathcal{Y}) - \frac{1}{S}\sum_{t=T+1}^{T+S}u_t^h(\mathcal{Y})\right\rangle\right] \\
\Rightarrow |\mathbf{R}_{te}(h) - \mathbf{R}_{tr}(h)| &\le \frac{1}{M}\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim \mathrm{D}}\left[\left|\frac{1}{S}\sum_{t=T+1}^{T+S}\|u_t^h(\mathcal{Y})\|_2^2 - \frac{1}{T}\sum_{t=1}^{T}\|u_t^h(\mathcal{Y})\|_2^2\right|\right. \\
&\quad \left. + 2\|\mathcal{Y}-\mathcal{X}\|\left\|\frac{1}{T}\sum_{t=1}^{T}u_t^h(\mathcal{Y}) - \frac{1}{S}\sum_{t=T+1}^{T+S}u_t^h(\mathcal{Y})\right\|_2\right].
\end{aligned}
$$

Note that

$$
\begin{aligned}
&\left|\frac{1}{S}\sum_{t=T+1}^{T+S}\|u_t^h(\mathcal{Y})\|^2 - \frac{1}{T}\sum_{t=1}^{T}\|u_t^h(\mathcal{Y})\|^2\right| \\
\le &\frac{1}{\varepsilon^2}\underbrace{\left|\frac{1}{S}\sum_{t=T+1}^{T+S}\|\nabla R_t(\mathcal{Y})\|^2 - \frac{1}{T}\sum_{t=1}^{T}\|\nabla R_t(\mathcal{Y})\|^2\right|}_{A_2}
\end{aligned}
$$

and

$$\begin{aligned}&\|\mathcal{Y}-\mathcal{X}\|\left\|\frac{1}{T}\sum_{t=1}^{T}u_t^h(\mathcal{Y})-\frac{1}{S}\sum_{t=T+1}^{T+S}u_t^h(\mathcal{Y})\right\|\\ \leq&\frac{(B_d+1)\sqrt{M}}{\varepsilon}\underbrace{\left\|\frac{1}{T}\sum_{t=1}^{T}\nabla R_t(\mathcal{Y})-\frac{1}{S}\sum_{t=T+1}^{T+S}\nabla R_t(\mathcal{Y})\right\|}_{A_1}.\end{aligned}$$

Then we have

$$\begin{aligned}|\mathbf{R}_{te}(h)-\mathbf{R}_{tr}(h)|\leq&\frac{1}{M}\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim\mathrm{D}}\left[\frac{4B_d\sqrt{M}}{\varepsilon}A_1+\frac{1}{\varepsilon^2}A_2\right]\\ =&\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim\mathrm{D}}\left[\frac{4B_d}{\sqrt{M}\varepsilon}A_1+\frac{1}{M\varepsilon^2}A_2\right].\end{aligned}$$

The proof is then completed. ∎

Proof of Theorem 3.9.

$$\begin{aligned}\mathbf{E}_g=&\mathbf{R}_{te}(\hat{h}_{tr})-\mathbf{R}_{te}(h_0^*)\\ =&\underbrace{\mathbf{R}_{te}(\hat{h}_{tr})-\mathbf{R}_{tr}(\hat{h}_{tr})}_{(a)}+\underbrace{\mathbf{R}_{tr}(\hat{h}_{tr})-\mathbf{R}_{tr}(h_{tr}^*)}_{(b)}+\underbrace{\mathbf{R}_{tr}(h_{tr}^*)-\mathbf{R}_{tr}(h_0^*)}_{(c)}+\underbrace{\mathbf{R}_{tr}(h_0^*)-\mathbf{R}_{te}(h_0^*)}_{(d)}.\end{aligned}$$

According to the definition of h_{tr}^*, we get $(c)\leq 0$. (b) is the training error that we have estimated as in Theorem 3.7. And by Lemma 3.8, we can directly get

$$(a)+(c)\leq\mathbb{E}_{(\mathcal{Y},\mathcal{X})\sim\mathrm{D}}\left[\frac{8B_d}{\sqrt{M}\varepsilon}A_1+\frac{2}{M\varepsilon^2}A_2\right].$$

Then, the final result is achieved. ∎

Appendix B. Details on solving DLW-models. We present the details of ADMM to solve the three basic models, i.e. DLW-NUCLR, DLW-TV, DLW-TVS, and three different target models, i.e. DLW-LRTV, DLW-E3DTV, and DLW-LRTFDFR.

B.1. DLW-NN. The DLW-NN aims to solve the following problem

$$\min_X\ \frac{1}{2}\|W\odot(Y-X)\|^2+\lambda\|X\|_*, \tag{B.1}$$

where $W,X,Y\in\mathbb{R}^{hw\times b}$. We use the ADMM algorithm to solve it. Firstly, the problem can be equivalently written as

$$\begin{aligned}&\min_{X,Z}\ \frac{1}{2}\|W\odot(Y-Z)\|^2+\lambda\|X\|_*,\\ &\text{s.t. } X=Z.\end{aligned}$$

The corresponding augmented Lagrange function is

$$\mathcal{L} = \frac{1}{2}\|W \odot (Y - Z)\|^2 + \lambda\|X\|_* + \frac{\mu}{2}\|X - Z\|^2 + \langle \Gamma, X - Z\rangle,$$

where Γ is the Lagrange multiplier. Then we iteratively solve the problem (B.1). Specifically, in the k-th step, the update rules are as follows.

1) update Z: The Z-subproblem is

$$\min_Z \ \frac{1}{2}\|W \odot (Y - Z)\|^2 + \frac{\mu}{2}\left\|X^{(k-1)} - Z + \frac{\Gamma^{(k-1)}}{\mu}\right\|^2.$$

This subproblem has a closed-form solution

$$Z^{(k)} = \frac{W^2 \odot Y + \mu X^{(k-1)} + \Gamma^{(k-1)}}{W^2 + \mu}.$$

2) update X: The X-subproblem is

$$\min_X \ \lambda\|X\|_* + \frac{\mu}{2}\left\|X - Z^{(k)} + \frac{\Gamma^{(k-1)}}{\mu}\right\|^2.$$

This subproblem also has a closed-form solution

$$X^{(k)} = U^{(k)}\text{Shrink}\left(\Sigma^{(k)}, \frac{\lambda}{\mu}\right)V^{(k)T}.$$

$U^{(k)}\Sigma^{(k)}V^{(k)T}$ is the singular value decomposition (SVD) of $Z^{(k)} - \frac{\Gamma^{(k-1)}}{\mu}$. The shrinkage operator $\text{Shrink}(\cdot,\cdot)$ is defined as

$$\text{Shrink}(X, \eta) = \begin{cases} X - \eta, & X \geq \eta; \\ 0, & -\eta \leq X \leq \eta; \\ X + \eta, & X \leq -\eta. \end{cases}$$

3) update Γ:

$$\Gamma^{(k)} = \Gamma^{(k-1)} + \mu(X^{(k)} - Z^{(k)}).$$

B.2. DLW-TV. The DLW-TV aims to solve the following problem

$$\min_{\mathcal{X}} \ \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{X})\|^2 + \lambda\|\mathcal{X}\|_{TV}, \tag{B.2}$$

where $\mathcal{W}, \mathcal{Y}, \mathcal{X} \in \mathbb{R}^{h\times w\times b}$. We use the ADMM algorithm to solve it. Firstly, the problem can be equivalently written as

$$\begin{aligned} &\min_{\mathcal{X},\mathcal{Z}} \ \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{Z})\|^2 + \lambda\left(\|\mathcal{X}_1\|_1 + \|\mathcal{X}_2\|_1\right), \\ &\text{s.t. } \mathcal{X}_1 = \nabla_1\mathcal{Z}, \mathcal{X}_2 = \nabla_2\mathcal{Z}, \end{aligned}$$

where ∇_i is the difference operator along the i-th dimension. Specifically,

$$\|\nabla_1 \mathcal{Z}\|_1 := \sum_{i,j,n} |\mathcal{Z}(i+1,j,n) - \mathcal{Z}(i,j,n)|,$$
$$\|\nabla_2 \mathcal{Z}\|_1 := \sum_{i,j,n} |\mathcal{Z}(i,j+1,n) - \mathcal{Z}(i,j,n)|.$$

The corresponding augmented Lagrange function is

$$\begin{aligned}\mathcal{L} =& \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{Z})\|^2 + \lambda \left(\|\mathcal{X}_1\|_1 + \|\mathcal{X}_2\|_1\right) + \\ & \frac{\mu}{2}\left(\|\mathcal{X}_1 - \nabla_1 \mathcal{Z}\|^2 + \langle \Gamma_1, \mathcal{X}_1 - \nabla_1 \mathcal{Z}\rangle + \|\mathcal{X}_2 - \nabla_2 \mathcal{Z}\|^2 + \langle \Gamma_2, \mathcal{X}_2 - \nabla_2 \mathcal{Z}\rangle\right),\end{aligned}$$

where Γ_1 and Γ_2 are the Lagrange multipliers. Then we iteratively solve the problem (B.2). Specifically, in the k-th step, the updating rules are as follows.

1) update $\mathcal{Z}$: The $\mathcal{Z}$-subproblem is

(B.3)

$$\min_{\mathcal{Z}} \ \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{Z})\|^2 + \frac{\mu}{2}\left(\left\|\mathcal{X}_1^{(k-1)} - \nabla_1 \mathcal{Z} + \frac{\Gamma_1^{(k-1)}}{\mu}\right\|^2 + \left\|\mathcal{X}_2^{(k-1)} - \nabla_2 \mathcal{Z} + \frac{\Gamma_2^{(k-1)}}{\mu}\right\|^2\right).$$

Let Z_n denote the n-th slice, $\mathcal{Z}(:,:,n)$, of $\mathcal{Z}$, and then (B.3) can be split into b equivalent problems, and each has closed-form solution

$$\begin{aligned} Z_n^{(k)} &= \mathbf{F}^{-1}\left(\frac{A_{1n} + (W_n^2/\mu) \odot \mathbf{F}(Y_n)}{A_{2n} + W_n^2/\mu}\right), n = 1, ..., b, \\ \text{where } A_{1n} &= \mathbf{F}(\nabla_1)^* \odot \mathbf{F}(w_{1n}) + \mathbf{F}(\nabla_2)^* \odot \mathbf{F}(w_{2n}), \\ A_{2n} &= \mathbf{F}(\nabla_1)^* \odot \mathbf{F}(\nabla_1) + \mathbf{F}(\nabla_2)^* \odot \mathbf{F}(\nabla_2), \\ w_{1n} &= X_{1n}^{(k-1)} + \frac{\Gamma_{1n}^{(k-1)}}{\mu}, \\ w_{2n} &= X_{2n}^{(k-1)} + \frac{\Gamma_{2n}^{(k-1)}}{\mu}. \end{aligned}$$

$\mathbf{F}$ and $\mathbf{F}^{-1}$ denote the discrete Fourier transform and its reverse, respectively. Finally, $\mathcal{Z}^{(k)}$ can be derived by aggregating its slices $Z_n^{(k)}$ for $n \in [b]$.

2) update $\mathcal{X}$: The $\mathcal{X}_l$-subproblem is

$$\min_{\mathcal{X}} \ \lambda\|\mathcal{X}\|_1 + \frac{\mu}{2}\left\|\mathcal{X} - \nabla_l \mathcal{Z} + \frac{\Gamma_l^{(k-1)}}{\mu}\right\|^2, \ l = 1, 2.$$

Each has a closed-form solution as

$$\mathcal{X}_l^{(k)} = \text{Shrink}\left(\nabla_l \mathcal{Z} - \frac{\Gamma_l^{(k-1)}}{\mu}, \frac{\lambda}{\mu}\right), \ l = 1, 2.$$

3) update Γ:

$$\Gamma_1^{(k)} = \Gamma_1^{(k-1)} + \mu(\mathcal{X}_1^{(k)} - \nabla_1 \mathcal{Z}^{(k)}),$$
$$\Gamma_2^{(k)} = \Gamma_2^{(k-1)} + \mu(\mathcal{X}_2^{(k)} - \nabla_2 \mathcal{Z}^{(k)}).$$

B.3. DLW-TVS. The DLW-TVS aims to solve the following problem

$$\min_{\mathcal{X}} \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{X})\|^2 + \lambda \|\mathcal{X}\|_{TVS}, \tag{B.4}$$

where $\mathcal{W}, \mathcal{Y}, \mathcal{X} \in \mathbb{R}^{h \times w \times b}$. We use the ADMM algorithm to solve it. Firstly, the problem can be equivalently written as

$$\min_{\mathcal{X},\mathcal{Z}} \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{Z})\|^2 + \lambda \|\mathcal{X}\|_1,$$
$$\text{s.t. } \mathcal{X} = \nabla_3 \mathcal{Z},$$

where ∇_3 is the difference operator along the third dimension. Specifically,

$$\|\nabla_3 \mathcal{Z}\|_1 := \sum_{i,j,n} |\mathcal{Z}(i,j,n+1) - \mathcal{Z}(i,j,n)|.$$

The corresponding augmented Lagrange function is

$$\mathcal{L} = \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{Z})\|^2 + \lambda \|\mathcal{X}\|_1 + \frac{\mu}{2}\|\mathcal{X} - \nabla_3 \mathcal{Z}\|^2 + \langle \Gamma, \mathcal{X} - \nabla_3 \mathcal{Z} \rangle.$$

Then we iteratively solve the problem (B.4). Specifically, in the k-th step, the updating rules are as follows.

1) update $\mathcal{Z}$: The $\mathcal{Z}$-subproblem is

$$\min_{\mathcal{Z}} \frac{1}{2}\|\mathcal{W} \odot (\mathcal{Y} - \mathcal{Z})\|^2 + \frac{\mu}{2}\left(\left\|\mathcal{X}^{(k-1)} - \nabla_3 \mathcal{Z} + \frac{\Gamma^{(k-1)}}{\mu}\right\|^2\right).$$

Let $Z_{(3)} \in \mathbb{R}^{hw \times b}$ denote the third-mode unfolding of $\mathcal{Z}$, that is

$$\mathcal{Z}(i,j,:) = Z_{(3)}((i-1)w + j, :).$$

Then the subproblem has a closed-form solution as

$$Z_{(3)}^{(k)} = \mathbf{F}^{-1}\left(\frac{A_1 + (W_{(3)}^2/\mu) \odot \mathbf{F}(Y_{(3)})}{A_2 + W_{(3)}^2/\mu}\right),$$
$$\text{where } A_1 = \mathbf{F}(\nabla_2)^* \odot \mathbf{F}(w),$$
$$A_2 = \mathbf{F}(\nabla_2)^* \odot \mathbf{F}(\nabla_2),$$
$$w = X_{(3)}^{(k-1)} + \frac{\Gamma_{(3)}^{(k-1)}}{\mu}.$$

Finally, $\mathcal{Z}^{(k)}$ can be derived by folding $Z_{(3)}^{(k)}$ along the third mode.

2) update $\mathcal{X}$: The $\mathcal{X}$-subproblem is

$$\min_{\mathcal{X}} \ \lambda\|\mathcal{X}\|_1 + \frac{\mu}{2}\left\|\mathcal{X} - \nabla_3\mathcal{Z} + \frac{\Gamma^{(k-1)}}{\mu}\right\|^2.$$

Its solution is

$$\mathcal{X}^{(k)} = \text{Shrink}\left(\nabla_3\mathcal{Z} - \frac{\Gamma^{(k-1)}}{\mu}, \frac{\lambda}{\mu}\right).$$

3) update Γ:

$$\Gamma^{(k)} = \Gamma^{(k-1)} + \mu(\mathcal{X}^{(k)} - \nabla_3\mathcal{Z}^{(k)}).$$

B.4. DLW-LRTV. The LRTV [15] method solves the following problem

$$\begin{aligned}\min_{X,S\in\mathbb{R}^{hw\times b}} \ &||X||_* + \tau\|X\|_{HTV} + \lambda\|S\|_1,\\ \text{s.t. } &\|Y - X - S\|^2 \le \varepsilon, \text{rank}(X) \le r.\end{aligned}$$

We derive our DLW-LRTV by replacing $\|S\|_1$ with our weighted loss term, and the problem is formulated as

$$\begin{aligned}\min_{X\in\mathbb{R}^{hw\times b}} \ &\frac{\kappa}{2}\|W\odot(Y-X)\|^2 + ||X||_* + \tau\|X\|_{HTV},\\ \text{s.t. } &\text{rank}(X) \le r.\end{aligned} \tag{B.5}$$

Similar to LRTV, we also use the ADMM algorithm to solve our DLW-LRTV. Firstly, (B.5) can be equivalently written as

$$\begin{aligned}\min_{X,Z,L} \ &\frac{\kappa}{2}\|W\odot(Y-Z)\|^2 + ||L||_* + \tau\|X\|_{HTV},\\ \text{s.t. } &\text{rank}(L) \le r, L = Z, X = Z.\end{aligned}$$

Its augmented Lagrange function is

$$\begin{aligned}\mathcal{L} =&\frac{\kappa}{2}\|W\odot(Y-Z)\|^2 + \|L\|_* + \tau\|X\|_{HTV} + \frac{\mu_1}{2}\|L-Z\|^2 + \langle\Gamma_1, L-Z\rangle +\\ &\frac{\mu_2}{2}\|X-Z\|^2 + \langle\Gamma_2, X-Z\rangle.\end{aligned}$$

Then the iterative updating rules are:

1) update L: The L-subproblem is

$$\begin{aligned}\min_{L} \ &\|L\|_* + \frac{\mu_1}{2}\left\|L - Z^{(k-1)} + \frac{\Gamma_1^{(k-1)}}{\mu_1}\right\|^2,\\ \text{s.t. } &\text{rank}(L) \le r.\end{aligned}$$

Its solution is

$$L^{(k-1)} = U^{(k-1)}\text{Shrink}\left(\Sigma_r^{(k-1)}, \frac{1}{\mu_1}\right)V^{(k-1)T},$$

where $U^{(k-1)}\Sigma^{(k-1)}V^{(k-1)T}$ is SVD of $Z^{(k-1)} - \frac{\Gamma_1^{(k-1)}}{\mu_1}$. The first r diagonal elements of Σ_r are the same as Σ while the remaining elements are all 0.

2) update X**:**The X-subproblem is

$$\min_X \ \tau\|X\|_{HTV} + \frac{\mu_2}{2}\left\|X - Z^{(k-1)} + \frac{\Gamma_2^{(k-1)}}{\mu_2}\right\|^2.$$

We inherit the fast gradient-based algorithm for solving this subproblem from the original LRTV. Details can be seen in [15].

3) update Z**:** The Z-subproblem is

$$\begin{aligned}\min_Z \ \frac{\kappa}{2}\|W \odot (Y - Z)\|^2 + \frac{\mu_1}{2}\left\|L^{(k)} - Z + \frac{\Gamma_1^{(k-1)}}{\mu_1}\right\|^2 \\ + \frac{\mu_2}{2}\left\|X^{(k)} - Z + \frac{\Gamma_2^{(k-1)}}{\mu_2}\right\|^2.\end{aligned}$$

It has a closed-form solution

$$\begin{aligned}Z^{(k)} &= \frac{\kappa W^2 \odot Y + A}{\kappa W^2 + \mu_1 + \mu_2}, \\ \text{where } A &= \mu_1 L^{(k)} + \mu_2 X^{(k)} + \Gamma_1^{(k-1)} + \Gamma_2^{(k-1)}.\end{aligned}$$

4) update Γ**:**

$$\begin{aligned}\Gamma_1^{(k)} &= \Gamma_1^{(k-1)} + \mu(L^{(k)} - Z^{(k)}), \\ \Gamma_2^{(k)} &= \Gamma_2^{(k-1)} + \mu(X^{(k)} - Z^{(k)}).\end{aligned}$$

The parameter τ and κ are set as 0.001 and 0.14, respectively, for all experiments. The rank r for CAVE and ICVL is set as 3, and 5 for PaviaU, DCmall, Indian Pine and Urban datasets.

B.5. DLW-E3DTV.

The E3DTV method [32] solves the following problem

$$\begin{aligned}\min_{\mathcal{X},U_n,V_n,\mathcal{E}} \ &\sum_{n=1}^{3}\tau\|U_n\|_1 + \|\mathcal{E}\|_1, \\ \text{s.t. } &\mathcal{Y} = \mathcal{X} + \mathcal{E}, \ \nabla_n\mathcal{X} = U_nV_n^T, V_n^TV_n = I, \\ &V_n \in \mathbb{R}^{b\times r}, U_n \in \mathbb{R}^{hw\times r}, n = 1,2,3.\end{aligned}$$

And we formulate our DLW-E3DTV as

$$\begin{aligned}\min_{\mathcal{X},U_n,V_n,\mathcal{E}} \ & \frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{Z})\|^2+\sum_{n=1}^{3}\tau\|U_n\|_1,\\ \text{s.t. } & \mathcal{X}=\mathcal{Z},\ \nabla_n\mathcal{X}=U_nV_n^T, V_n^TV_n=I,\\ & V_n\in\mathbb{R}^{b\times r}, U_n\in\mathbb{R}^{hw\times r}, n=1,2,3.\end{aligned} \tag{B.6}$$

Similar to E3DTV, we also apply the ADMM algorithm to solve (B.6). The corresponding augmented Lagrange function is

$$\begin{aligned}\mathcal{L}=&\frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{Z})\|^2+\sum_{n=1}^{3}\tau\|U_n\|_1+\frac{\mu}{2}\|\mathcal{X}-\mathcal{Z}\|^2+\langle\Gamma_4,\mathcal{X}-\mathcal{Z}\rangle\\ &+\frac{\mu}{2}\sum_{n=1}^{3}\left\{\|\nabla_n\mathcal{X}-U_nV_n^T\|^2+\langle\Gamma_n,\nabla_n\mathcal{X}-U_nV_n^T\rangle\right\}.\end{aligned}$$

We need to iteratively solve the subproblems about $\mathcal{X},\mathcal{Z},U_n,V_n$. Note that for $\mathcal{X},U_n,V_n$ subproblems are exactly the same type as in the original E3DTV algorithm, and we only have to change the corresponding inputs. Please refer to [32] for details. In the k-th iteration, the $\mathcal{Z}$-subproblem is

$$\min_{\mathcal{Z}}\ \frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{Z})\|^2+\frac{\mu}{2}\left\|\mathcal{X}^{(k-1)}-\mathcal{Z}+\frac{\Gamma_4^{(k-1)}}{\mu}\right\|^2.$$

It has a closed-form solution

$$\mathcal{Z}^{(k)}=\frac{\mathcal{W}^2\odot\mathcal{Y}+\mu\mathcal{X}^{(k-1)}+\Gamma_4^{(k-1)}}{\mathcal{W}^2+\mu}.$$

The trade-off parameter τ is set as $10^{-4}*\sqrt{h*w}$ for CAVE, ICVL, PaviaU and DCmall dataset, and $10^{-4}*5*\sqrt{h*w}$ for Indian Pine and Urban datasets. The tensor rank for CAVE and ICVL datasets is set as [4,4,4] and [5,5,5] for PaviaU, DCmall, Indian Pines and Urban datasets.

B.6. DLW-LRTFDFR. The LRTFDFR method [49] solves the following problem

$$\min_{A,\mathcal{B},\mathcal{S}}\ \frac{1}{2}\|\mathcal{Y}-\mathcal{B}\times_3 A-\mathcal{S}\|^2+\tau\sum_{k=1}^{2}\|\mathcal{W}_k\odot(\mathcal{B}\times_k D_k)\|_{2,1}+\lambda\|D_3A\|^2+\mu\|\mathcal{W}_s\odot\mathcal{S}\|_1.$$

And we derive our DLW-LRTFDFR as

$$\min_{A,\mathcal{B}}\ \frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{B}\times_3 A)\|^2+\tau\sum_{k=1}^{2}\|\mathcal{W}_k\odot(\mathcal{B}\times_k D_k)\|_{2,1}+\lambda\|D_3A\|^2, \tag{B.7}$$

where $\mathcal{W} = h_\theta(\mathcal{Y})$ is the predicted weight. The original PAM-based algorithm can not conveniently solve DLW-LRTFDFR. Thus, we instead use the ADMM algorithm. Again, the problem (B.7) can be equivalently written as

$$\begin{aligned}&\min_{A,\mathcal{B},\mathcal{Z}} \frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{Z})\|^2+\tau\sum_{k=1}^{2}\|\mathcal{W}_k\odot(\mathcal{B}\times_k D_k)\|_{2,1}+\lambda\|D_3A\|^2,\\ &\text{s.t. } \mathcal{Z}=\mathcal{B}\times_3 A.\end{aligned}$$

The corresponding augmented Lagrange function is

$$\begin{aligned}\mathcal{L}=&\frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{Z})\|^2+\tau\sum_{k=1}^{2}\|\mathcal{W}_k\odot(\mathcal{B}\times_k D_k)\|_{2,1}+\lambda\|D_3A\|^2+\\ &\frac{\mu}{2}\|\mathcal{Z}-\mathcal{B}\times_3 A\|^2+\langle\Gamma,\mathcal{Z}-\mathcal{B}\times_3 A\rangle.\end{aligned}$$

Note that the subproblems about $\mathcal{B}, A$ are the same type as in the original PAM-based algorithm when solving LRTFDFR. Thus we also directly inherit the corresponding parts from the original algorithm. In the k-th iteration, the $\mathcal{Z}$-subproblem is

$$\min_{\mathcal{Z}} \frac{1}{2}\|\mathcal{W}\odot(\mathcal{Y}-\mathcal{Z})\|^2+\frac{\mu}{2}\left\|\mathcal{Z}-\mathcal{B}^{(k-1)}\times_3 A^{(k-1)}+\frac{\Gamma^{(k-1)}}{\mu}\right\|^2.$$

Its closed-form solution is

$$\mathcal{Z}^{(k)}=\frac{\mathcal{W}^2\odot\mathcal{Y}+\mu\mathcal{B}^{(k-1)}\times_3 A^{(k-1)}-\Gamma^{(k-1)}}{\mathcal{W}^2+\mu}.$$

The trade-off parameters τ and λ are set as 0.01 and 0.1, respectively, for all experiments. The rank r is set as 3 for the CAVE and ICVL datasets, 5 for the PaviaU and DCmall datasets, and 6 for the Indian Pine and Urban datasets.

Appendix C. Additional denoising Results. Six datasets are used for testing the performance of the proposed weighting scheme. We extract part of the original DCmall dataset to fine-tune the DLWnet trained on the CAVE dataset. The rest of DCmall with size $200 \times 200 \times 152$ is used for testing. The fine-tuned DLWnet is used for PaviaU, DCmall, Indian Pine, and Urban datasets. Table 8, Table 9 and Table 10 show the quantitative denoising comparison on CAVE, PaviaU and DCmall datasets, respectively, for all the advanced complex noise removal methods and *our* methods. We find that some methods may achieve very high performance on some images and noise patterns. However, it can be observed that our method performs more robust across different images with diverse noise patterns, and can also gain superior denoising results in most cases.

Table 11, Table 12 and Table 13 show the quantitative denoising comparison for the three basic models, respectively, with different DLWnets. Basically, we can see that when the source and target models match, the denoising performances are the best. For some images, it is also seen that DLWnet trained with combined source models gain the best denoising results. For

Table 8: Average test performance of different denoising competing methods on the CAVE dataset. The best results in each **column** are in **bold**, and the second best results in each **column** are with underline.

noise →	Case 1	Case 2	Case 3	Case 4	Case 5
index →	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
LRMR	24.48/0.6488	28.15/0.7314	26.92/0.713	26.45/0.6873	23.2/0.6083
LRTV	30.88/0.8887	31.75/0.9046	30.86/0.8907	32.68/0.9246	30.49/0.8832
NMoG	23.63/0.6713	28.1/0.817	26.79/0.8009	24.97/0.718	20.21/0.5506
HyRes	28.39/0.7901	34.04/0.9247	31.86/0.907	31.99/0.899	26.02/0.7512
FastHyMix	27.95/0.7949	33.39/0.9111	31.96/0.9208	33.13/0.9135	26.37/0.7657
CTV-RPCA	30.87/0.7898	30.38/0.7648	30.42/0.7815	28.37/0.6851	29.06/0.736
E3DTV	33.71/ **0.9218**	33.21/0.9099	32.98/0.9061	31.91/0.8822	31.6/ 0.8835
LRTFDFR	26.88/0.7848	26.28/0.752	25.93/0.7521	26.39/0.7597	25.21/0.7204
HSI-DeNet	27.59/0.7028	26.84/0.6311	26.9/0.6329	27.88/0.7182	28.4/0.6949
HSI-CNN	**34.04**/0.8607	**37.13**/**0.9552**	**35.44**/**0.9521**	**38.88**/**0.9661**	32.19/0.8604
DLW-LRTV	31.94/0.8851	32.78/0.9145	32.2/0.9045	33.11/0.9233	31.55/ **0.8930**
DLW-E3DTV	33.98/0.9056	34.66/0.9392	34.51/0.9369	33.96/0.9257	**32.55**/0.8823
DLW-LRTFDFR	31.32/0.8831	31.82/0.9042	31.57/0.8936	31.99/0.9039	30.96/0.8787

example, in Table 11, for Case 1 part of target model 1, type-'N+TS' DLWnet generalizes better than type-'N' DLWnet. However, it can be explored that the TVS regularization that describes the spectral smoothness property has a relationship with the spectral low-rank property, and thus the corresponding DLWnets may perform in a relatively similar way. For some images, type-'N+TS' DLWnet is possible to surpass type-'N' DLWnet for target model DLW-NN. This is not contradictory to their overall generalization performance.

Table 14, Table 15 and Table 16 show the quantitative denoising results on CAVE, PaviaU and DCmall datasets, respectively, obtained by LRTV, E3DTV, LRTFDFR and our corresponding weighted models with different DLWnets. Generally, our weighted models outperform the original models for most images. Generally, the type-'N+T+TS' DLWnet performs robustly and thus we advise first applying this DLWnet to a new noisy image.

Figure 8 and Figure 9 show the visual comparison of the CAVE and Indian Pines datasets, respectively, calculated by advanced denoising methods and our methods. For the Indian Pine datasets, our DLW-E3DTV can obtain better visual results although the noisy band is severely polluted. Figure 10 and Figure 11 show the visual comparison of the Indian Pine and Urban datasets, respectively, obtained by LRTV, E3DTV, LRTFDFR and our corresponding weighted models. It can be easily observed that most DLWnets can help the denoising models to remove complex noise better.

REFERENCES

[1] B. Arad and O. Ben-Shahar, *Sparse recovery of hyperspectral signal from natural rgb images*, in European Conference on Computer Vision, Springer, 2016, pp. 19–34.

[2] P. L. Bartlett and S. Mendelson, *Rademacher and gaussian complexities: Risk bounds and structural results*, Journal of Machine Learning Research, 3 (2002), pp. 463–482.

[3] A. Beck, *First-order methods in optimization*, SIAM, 2017.

Table 9: Average test performance of different denoising competing methods on the PaviaU dataset. The best results in each **column** are in **bold**, and the second best results in each **column** are with underline.

noise →	Case 1	Case 2	Case 3	Case 4	Case 5
index →	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
LRMR	25.05/0.5965	28.43/0.7108	27.99/0.7091	27.01/0.654	24.24/0.5628
LRTV	32.61/0.8793	33.1/0.8847	31.81/0.8687	34.91/0.9159	31.69/0.8792
NMoG	30.53/0.8077	32.99/0.8636	31.41/0.8407	28.97/0.7502	14.9/0.1078
HyRes	30.37/0.8523	34.40/0.9214	33.32/0.9166	31.19/0.8469	28.32/0.8065
FastHyMix	31.89/0.8899	**35.55**/**0.9349**	**33.68**/0.898	33.24/0.9046	29.09/0.8258
CTV-RPCA	30.33/0.7887	30.52/0.7954	29.75/0.7774	28.56/0.7247	28.05/0.7224
E3DTV	32.78/0.8656	33.06/0.876	32.33/0.8663	31.2/0.8345	30.89/0.8383
LRTFDFR	20.54/0.4496	20.69/0.4503	20.5/0.4488	20.68/0.4505	20.23/0.4464
HSI-DeNet	29.44/0.8194	30.78/0.846	30.48/0.8388	32.22/0.8932	29.94/0.8322
HSI-CNN	32.72/0.8972	33.84/0.9158	32.79/0.9046	35.24/0.934	31.4/0.8904
DLW-LRTV	33.67/0.9076	34.11/0.9143	33.2/0.9081	**35.84**/0.9348	33.21/0.9139
DLW-E3DTV	**33.77**/**0.9189**	34/0.9244	33.60/**0.9203**	32.01/0.8933	32.13/0.8939
DLW-LRTFDFR	33.63/0.9170	33.64/0.9169	33.16/0.9109	35.79/**0.9491**	**33.84**/**0.9289**

Table 10: Average test performance of different denoising competing methods on the DC dataset. The best results in each **column** are in **bold**, and the second best results in each **column** are with underline.

noise →	Case 1	Case 2	Case 3	Case 4	Case 5
index →	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
LRMR	26.07/0.7463	30.42/0.8676	29.78/0.8597	29.18/0.8372	25.59/0.734
LRTV	28.1/0.855	31.87/0.9093	29.75/0.8911	34.15/0.942	28.56/0.8663
NMoG	24.23/0.72	35.81/0.953	34.87/0.9549	31.61/0.9105	19.5/0.4763
HyRes	32.3/0.8779	**37.11**/**0.9666**	35.53/0.9549	33.16/0.9223	30/0.858
FastHyMix	33.22/0.8942	37.04/0.9598	**36.13**/**0.9571**	34.61/0.9466	31.12/0.8831
CTV-RPCA	31.49/0.8781	31.57/0.8779	30.76/0.8627	29.33/0.8239	28.86/0.8175
E3DTV	32.94/0.9115	34.21/0.9293	32.97/0.9138	32.09/0.8961	30.82/0.8822
LRTFDFR	33.31/0.9238	35.2/0.9516	33.44/0.9333	**36.37**/**0.9625**	19.82/0.2832
HSI-DeNet	27.64/0.8047	28.62/0.8449	28.56/0.8384	29.65/0.8798	28.31/0.8237
HSI-CNN	31.12/0.8865	32.42/0.9145	31.66/0.9063	33.81/0.9326	30.48/0.8832
DLW-LRTV	30.43/0.8875	31.57/0.9122	29.98/0.896	33.69/0.9434	29/0.8794
DLW-E3DTV	**33.74**/**0.9417**	34.21/0.9497	33.68/0.9443	32.7/0.9224	32.50/0.9244
DLW-LRTFDFR	33.67/0.9399	33.87/0.9434	33.28/0.9358	35.44/0.9608	**33.73**/**0.9423**

[4] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, *A theory of learning from different domains*, Machine Learning, 79 (2010), pp. 151–175.

[5] G. Blanchard, A. A. Deshmukh, Ü. Dogan, G. Lee, and C. Scott, *Domain generalization by marginal transfer learning*, The Journal of Machine Learning Research, 22 (2021), pp. 46–100.

[6] G. Blanchard, G. Lee, and C. Scott, *Generalizing from several related classification tasks to a new unlabeled sample*, Advances in Neural Information Processing Systems, 24 (2011).

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning, 3 (2011), pp. 1–122.

Table 11: Average test performance on CAVE dataset by DLW-NN, DLW-TV and DLW-TVS. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

	index	N	T	TS	N+T	N+TS	T+TS	N+T+TS
target model 1 : **DLW-NN**								
Case 1	PSNR	31.53	22.83	26.86	31.45	**31.72**	27.24	31.63
	SSIM	0.8639	0.5529	0.7545	0.8697	0.8704	0.7529	**0.8705**
Case 2	PSNR	31.96	22.06	26.62	31.87	**32.57**	26.64	32.26
	SSIM	0.8984	0.5425	0.7411	0.8990	**0.9108**	0.7586	0.9090
Case 3	PSNR	31.58	22.42	26.36	31.38	**31.68**	26.72	31.56
	SSIM	0.9026	0.5660	0.7545	0.9003	0.9045	0.7711	**0.9053**
Case 4	PSNR	30.78	18.69	23.80	30.42	**30.99**	23.46	30.69
	SSIM	0.8675	0.4160	0.6227	0.8638	**0.8816**	0.6383	0.8774
Case 5	PSNR	**29.48**	20.33	24.00	29.29	29.17	24.57	29.22
	SSIM	0.8249	0.4519	0.6370	**0.8311**	0.8200	0.6596	0.8253
target model 2 : **DLW-TV**								
Case 1	PSNR	31.59	**33.87**	29.57	33.74	31.32	33.45	33.00
	SSIM	0.8472	0.8785	0.7788	**0.8928**	0.8434	0.8831	0.8749
Case 2	PSNR	32.14	**34.99**	29.91	34.41	31.78	33.99	33.74
	SSIM	0.8849	**0.9369**	0.7941	0.9273	0.8745	0.9184	0.9155
Case 3	PSNR	31.65	**34.08**	29.01	33.57	31.28	32.97	33.03
	SSIM	0.8816	**0.9322**	0.7873	0.9208	0.8699	0.9094	0.9085
Case 4	PSNR	31.25	**33.43**	28.93	33.34	30.97	32.91	32.79
	SSIM	0.8695	**0.9141**	0.7776	0.9115	0.8615	0.9013	0.9001
Case 5	PSNR	29.67	31.00	27.06	**31.29**	29.36	30.66	30.82
	SSIM	0.8138	0.8385	0.7088	**0.8667**	0.8029	0.8429	0.8429
target model 3 : **DLW-TVS**								
Case 1	PSNR	29.76	24.74	**32.28**	29.55	31.32	31.48	30.97
	SSIM	0.8032	0.6146	**0.8505**	0.8094	0.8366	0.8430	0.8344
Case 2	PSNR	29.87	24.35	**33.10**	29.43	31.81	31.98	31.30
	SSIM	0.8332	0.6363	**0.8950**	0.8339	0.8762	0.8853	0.8702
Case 3	PSNR	30.31	24.75	**32.67**	29.81	31.99	31.79	31.45
	SSIM	0.8497	0.6655	**0.8935**	0.8479	0.8821	0.8847	0.8766
Case 4	PSNR	28.29	21.62	**31.53**	27.78	30.33	30.50	29.83
	SSIM	0.7944	0.5310	**0.8647**	0.7918	0.8453	0.8541	0.8393
Case 5	PSNR	28.49	22.63	**30.19**	28.20	29.83	29.55	29.46
	SSIM	0.769	0.5313	**0.8102**	0.7744	0.7992	0.8000	0.7972

[8] X. Cao, Q. Zhao, D. Meng, Y. Chen, and Z. Xu, *Robust low-rank matrix factorization under general mixture noise distributions*, IEEE Transactions on Image Processing, 25 (2016), pp. 4677–4690.

[9] Y. Chang, L. Yan, H. Fang, S. Zhong, and W. Liao, *Hsi-denet: Hyperspectral image restoration via convolutional neural network*, IEEE Transactions on Geoscience and Remote Sensing, 57 (2018), pp. 667–682.

[10] J. Chen and M. K. Ng, *Color image inpainting via robust pure quaternion matrix completion: Error bound and weighted loss*, SIAM Journal on Imaging Sciences, 15 (2022), pp. 1469–1498.

[11] Y. Chen, X. Cao, Q. Zhao, D. Meng, and Z. Xu, *Denoising hyperspectral image with non-iid noise structure*, IEEE Transactions on Cybernetics, 48 (2017), pp. 1054–1066.

Table 12: Average test performance on PaviaU by DLW-NN, DLW-TV and DLW-TVS. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

	index	N	T	TS	N+T	N+TS	T+TS	N+T+TS
target model 1 : **DLW-NN**								
Case 1	PSNR	33.32	21.39	27.58	32.11	**33.52**	25.68	32.82
	SSIM	0.9184	0.4196	0.6443	0.8765	**0.9211**	0.5551	0.8853
Case 2	PSNR	33.95	22.06	28.91	32.44	**34.02**	26.25	33.34
	SSIM	0.9260	0.4415	0.7142	0.8839	**0.9278**	0.5882	0.8961
Case 3	PSNR	**33.17**	22.02	27.8	31.87	32.99	25.88	32.32
	SSIM	0.9207	0.4400	0.6865	0.8766	**0.9218**	0.5791	0.8871
Case 4	PSNR	**31.90**	18.55	24.95	30.22	31.83	22.53	30.76
	SSIM	0.8880	0.3077	0.5176	0.8220	**0.8896**	0.4133	0.8267
Case 5	PSNR	**31.47**	19.73	24.99	29.98	30.87	23.32	30.24
	SSIM	**0.8691**	0.3494	0.5386	0.8199	0.8580	0.4519	0.8225
target model 2 : **DLW-TV**								
Case 1	PSNR	25.42	**29.83**	25.07	28.34	25.12	28.57	27.83
	SSIM	0.6293	**0.8284**	0.6157	0.7569	0.6228	0.7656	0.7316
Case 2	PSNR	25.61	**30.06**	25.18	28.74	25.11	28.89	28.17
	SSIM	0.6344	**0.8378**	0.6191	0.7695	0.6234	0.7776	0.7440
Case 3	PSNR	25.38	**29.50**	24.77	28.31	25.08	28.30	27.73
	SSIM	0.6308	**0.8288**	0.6120	0.7628	0.6215	0.7676	0.7364
Case 4	PSNR	24.44	**28.31**	24.61	27.39	24.33	27.61	26.93
	SSIM	0.5836	**0.7757**	0.5955	0.7237	0.5889	0.7364	0.7011
Case 5	PSNR	24.54	**28.16**	24.00	27.12	24.45	27.13	26.63
	SSIM	0.5978	**0.7827**	0.5823	0.7269	0.5973	0.7299	0.6995
target model 3 : **DLW-TVS**								
Case 1	PSNR	30.52	22.00	**32.26**	29.61	32.07	30.68	31.28
	SSIM	0.8151	0.4514	**0.8504**	0.8008	0.8501	0.8293	0.8416
Case 2	PSNR	31.08	22.41	**32.71**	29.71	32.41	30.85	31.45
	SSIM	0.8281	0.4585	**0.8639**	0.8106	0.8612	0.8389	0.8488
Case 3	PSNR	31.29	22.40	32.04	29.86	**32.37**	30.43	31.31
	SSIM	0.8238	0.4566	0.8533	0.8058	**0.8573**	0.8299	0.8434
Case 4	PSNR	28.55	19.67	**30.15**	27.56	30.07	28.57	29.23
	SSIM	0.7608	0.352	0.8003	0.7388	**0.8048**	0.7659	0.7855
Case 5	PSNR	29.52	20.71	29.94	28.36	**30.60**	28.85	29.77
	SSIM	0.7729	0.3917	0.7994	0.7530	**0.8128**	0.7731	0.7975

[12] C. Cortes and M. Mohri, *Domain adaptation in regression*, in International Conference on Algorithmic Learning Theory, Springer, 2011, pp. 308–323.

[13] Y. Q. Dai Wenyuan, X. Guirong, and Y. Yong, *Boosting for transfer learning*, in International Conference on Machine Learning, 2007, pp. 193–200.

[14] P. Germain, A. Habrard, F. Laviolette, and E. Morvant, *A new pac-bayesian perspective on domain adaptation*, in International Conference on Machine Learning, PMLR, 2016, pp. 859–868.

[15] W. He, H. Zhang, L. Zhang, and H. Shen, *Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration*, IEEE Transactions on Geoscience and Remote Sensing, 54 (2015), pp. 178–188.

Table 13: Average test performance on DCmall by DLW-NN, DLW-TV and DLW-TVS. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

	index	N	T	TS	N+T	N+TS	T+TS	N+T+TS
target model 1 : **DLW-NN**								
Case 1	PSNR	**33.80**	22.22	26.65	32.08	33.20	26.59	32.65
	SSIM	**0.9477**	0.5679	0.6956	0.9197	0.9270	0.6985	0.9189
Case 2	PSNR	**34.86**	22.81	27.96	32.34	34.12	26.96	33.17
	SSIM	**0.9596**	0.5846	0.7559	0.9249	0.9429	0.7189	0.9293
Case 3	PSNR	**34.15**	22.90	27.60	31.94	33.28	26.74	32.41
	SSIM	**0.9530**	0.5873	0.7504	0.9198	0.9331	0.7143	0.9199
Case 4	PSNR	**32.90**	19.12	24.22	30.55	31.73	23.25	31.04
	SSIM	**0.9331**	0.4416	0.5863	0.8853	0.8944	0.5580	0.8804
Case 5	PSNR	**31.67**	20.19	24.10	30.27	30.79	23.93	30.49
	SSIM	**0.9121**	0.4814	0.5820	0.8843	0.8838	0.5838	0.8784
target model 2 : **DLW-TV**								
Case 1	PSNR	24.41	**28.42**	23.96	26.91	24.11	26.96	26.42
	SSIM	0.5259	**0.7987**	0.5155	0.6884	0.5178	0.6916	0.6556
Case 2	PSNR	24.60	**28.84**	24.14	27.29	24.23	27.30	26.76
	SSIM	0.5320	**0.8183**	0.5237	0.7083	0.5243	0.7116	0.6752
Case 3	PSNR	24.33	**28.37**	23.72	26.84	24.13	26.82	26.33
	SSIM	0.5272	**0.8066**	0.5084	0.6969	0.5192	0.6985	0.6625
Case 4	PSNR	23.70	**27.63**	23.62	26.39	23.56	26.43	25.94
	SSIM	0.4749	**0.7809**	0.4998	0.6638	0.4820	0.6694	0.6327
Case 5	PSNR	23.63	**27.18**	23.19	25.89	23.59	25.91	25.45
	SSIM	0.4853	**0.7639**	0.4814	0.6513	0.4856	0.6517	0.6162
target model 3 : **DLW-TVS**								
Case 1	PSNR	28.27	21.96	**29.99**	27.74	29.64	29.04	29.19
	SSIM	0.8246	0.5826	0.8469	0.8186	**0.8515**	0.8448	0.8492
Case 2	PSNR	28.72	22.21	**30.61**	27.88	30.08	29.22	29.44
	SSIM	0.8382	0.5767	**0.8701**	0.8256	0.8672	0.8553	0.8607
Case 3	PSNR	28.79	22.31	**30.26**	27.98	29.98	29.04	29.39
	SSIM	0.8368	0.5784	**0.8641**	0.8249	0.8641	0.8494	0.8576
Case 4	PSNR	26.82	19.82	**28.53**	26.19	28.17	27.67	27.74
	SSIM	0.7806	0.4719	0.8021	0.7642	**0.8067**	0.7958	0.7999
Case 5	PSNR	27.01	20.56	28.29	26.48	**28.34**	27.71	27.93
	SSIM	0.7793	0.5131	0.7971	0.773	**0.8106**	0.7996	0.8067

[16] T.-X. Jiang, L. Zhuang, T.-Z. Huang, X.-L. Zhao, and J. M. Bioucas-Dias, *Adaptive hyperspectral mixed noise removal*, IEEE Transactions on Geoscience and Remote Sensing, 60 (2021), pp. 1–13.

[17] L. S. Kalman and E. M. Bassett III, *Classification and material identification in an urban environment using hydice hyperspectral data*, in Imaging Spectrometry III, vol. 3118, SPIE, 1997, pp. 57–68.

[18] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[19] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, *Noise2noise: Learning image restoration without clean data*, in International Conference on Machine Learning, PMLR, 2018, pp. 2965–2974.

Table 14: Average test performance on CAVE by LRTV, E3DTV, LRTFDFR and their corresponding DLW-models. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

	index	*Ori.	N	T	TS	N+T	N+TS	T+TS	N+T+TS
target model 1 : **LRTV**									
Case 1	PSNR	30.88	31.73	29.03	29.83	**32.09**	31.55	31.41	31.94
	SSIM	0.8887	0.8761	0.7534	0.8619	**0.8934**	0.8733	0.8763	0.8851
Case 2	PSNR	31.75	32.70	29.90	30.89	32.74	32.54	32.23	**32.78**
	SSIM	0.9046	0.9114	0.8163	0.8819	**0.9163**	0.9073	0.9071	0.9145
Case 3	PSNR	30.86	32.17	29.68	29.94	32.18	31.99	31.51	**32.20**
	SSIM	0.8907	0.9015	0.8199	0.8719	**0.9048**	0.8955	0.8973	0.9045
Case 4	PSNR	32.68	33.15	31.52	31.48	**33.16**	32.94	32.63	33.11
	SSIM	0.9246	0.9236	0.8864	0.9019	**0.9258**	0.918	0.918	0.9233
Case 5	PSNR	30.49	31.33	29.18	28.96	**31.61**	31.26	30.85	31.55
	SSIM	0.8832	0.8814	0.7904	0.8560	**0.8980**	0.8834	0.8838	0.8930
target model 2 : **E3DTV**									
Case 1	PSNR	33.71	33.46	31.09	33.11	33.97	33.61	**34.17**	33.98
	SSIM	**0.9218**	0.8932	0.8222	0.8961	0.9101	0.8991	0.9081	0.9056
Case 2	PSNR	33.21	34.32	31.38	33.96	34.49	34.48	**34.95**	34.66
	SSIM	0.9099	0.9314	0.8651	0.9244	0.9403	0.9354	**0.9423**	0.9392
Case 3	PSNR	32.98	34.28	31.4	33.06	34.3	34.36	34.44	**34.51**
	SSIM	0.9061	0.9326	0.8670	0.9115	**0.9386**	0.9331	0.9361	0.9369
Case 4	PSNR	31.91	33.51	29.33	33.13	33.64	33.67	**34.13**	33.96
	SSIM	0.8822	0.9167	0.7990	0.9119	0.9256	0.9199	**0.9278**	0.9257
Case 5	PSNR	31.60	32.31	28.86	30.99	32.44	32.37	32.22	**32.55**
	SSIM	0.8835	0.8758	0.7488	0.8616	**0.8912**	0.8757	0.8791	0.8823
target model 3 : **LRTFDFR**									
Case 1	PSNR	26.88	30.73	28.75	29.47	**31.39**	30.91	31.35	31.32
	SSIM	0.7848	0.8610	0.7879	0.8247	**0.8875**	0.8721	0.8865	0.8831
Case 2	PSNR	26.28	31.54	29.60	30.37	**31.83**	31.43	31.75	31.82
	SSIM	0.7520	0.8968	0.8464	0.8594	0.9034	0.8964	0.8995	**0.9042**
Case 3	PSNR	25.93	31.11	29.51	29.65	31.41	31.16	31.35	**31.57**
	SSIM	0.7521	0.8802	0.8451	0.8391	0.8902	0.8822	0.8891	**0.8936**
Case 4	PSNR	26.39	31.98	30.70	30.75	**32.19**	31.65	31.89	31.99
	SSIM	0.7597	0.8993	0.8924	0.8708	**0.9049**	0.8939	0.8961	0.9039
Case 5	PSNR	25.21	30.53	28.93	29.35	30.89	30.63	30.89	**30.96**
	SSIM	0.7204	0.8638	0.8069	0.8421	**0.8791**	0.8709	0.8820	0.8787

[20] F. Li, M. K. Ng, and C. Shen, *Multiplicative noise removal with spatially varying regularization parameters*, SIAM Journal on Imaging Sciences, 3 (2010), pp. 1–20.

[21] J. Li, Z. Shen, R. Yin, and X. Zhang, *A reweighted l2 method for image restoration with poisson and mixed poisson-gaussian noise*, Inverse Probl. Imaging (Springfield), 9 (2015), pp. 875–894.

[22] J. Liu, X.-C. Tai, H. Huang, and Z. Huan, *A weighted dictionary learning model for denoising images corrupted by mixed noise*, IEEE transactions on image processing, 22 (2012), pp. 1108–1120.

[23] Y.-S. Luo, X.-L. Zhao, T.-X. Jiang, Y.-B. Zheng, and Y. Chang, *Hyperspectral mixed noise removal via spatial-spectral constrained unsupervised deep image prior*, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14 (2021), pp. 9435–9449.

Table 15: Average test performance on PaviaU by LRTV, E3DTV, LRTFDFR and their corresponding DLW-models. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

	index	*Ori	N	T	TS	N+T	N+TS	T+TS	N+T+TS
target model 1 : **LRTV**									
Case 1	PSNR	32.61	32.25	26.87	30.72	33.34	31.94	33.52	**33.67**
	SSIM	0.8793	0.8886	0.6842	0.8522	**0.9105**	0.8793	0.9040	0.9076
Case 2	PSNR	33.10	33.14	26.87	31.55	33.63	32.38	33.88	**34.11**
	SSIM	0.8847	0.8981	0.6749	0.8725	0.9142	0.887	0.9135	**0.9143**
Case 3	PSNR	31.81	32.61	26.89	30.27	33.17	31.89	32.73	**33.20**
	SSIM	0.8687	0.8946	0.6813	0.8608	**0.9103**	0.8828	0.9073	0.9081
Case 4	PSNR	34.91	35.21	29.36	33.16	35.50	34.63	35.52	**35.84**
	SSIM	0.9159	0.9278	0.7933	0.9065	0.9376	0.9223	**0.9387**	0.9348
Case 5	PSNR	31.69	32.80	27.61	29.78	**33.48**	31.94	32.86	33.21
	SSIM	0.8792	0.9030	0.7329	0.8567	**0.9196**	0.8915	0.9096	0.9139
target model 2 : **E3DTV**									
Case 1	PSNR	32.78	32.79	26.64	32.18	33.14	32.57	33.73	**33.77**
	SSIM	0.8656	0.9069	0.6823	0.8912	0.9180	0.8996	**0.9190**	0.9189
Case 2	PSNR	33.06	33.62	26.27	33.26	32.96	33.04	33.86	**34.00**
	SSIM	0.876	0.9189	0.6555	0.9099	0.9168	0.9086	0.9231	**0.9244**
Case 3	PSNR	32.33	33.56	26.30	32.24	33.07	33.02	33.19	**33.60**
	SSIM	0.8663	0.9149	0.6547	0.9025	0.9159	0.9080	0.9176	**0.9203**
Case 4	PSNR	31.20	31.18	24.97	31.24	31.36	30.64	**32.22**	32.01
	SSIM	0.8345	0.8837	0.5964	0.8748	0.8878	0.8682	**0.8954**	0.8933
Case 5	PSNR	30.89	31.92	25.72	30.17	31.92	31.34	31.89	**32.13**
	SSIM	0.8383	0.8862	0.6385	0.8588	0.8913	0.8777	0.8901	**0.8939**
target model 3 : **LRTFDFR**									
Case 1	PSNR	20.54	32.75	24.71	33.32	32.26	33.43	33.36	**33.63**
	SSIM	0.4496	0.8926	0.5769	0.9108	0.8949	0.9102	0.9132	**0.9170**
Case 2	PSNR	20.69	33.47	24.68	**34.15**	32.27	33.78	33.32	33.64
	SSIM	0.4503	0.9015	0.5667	**0.9188**	0.8971	0.9153	0.9123	0.9169
Case 3	PSNR	20.50	33.01	24.66	32.56	32.03	**33.47**	32.58	33.16
	SSIM	0.4488	0.8930	0.5643	0.9039	0.8871	0.9106	0.9041	**0.9109**
Case 4	PSNR	20.68	36.21	26.98	**36.51**	34.26	**36.47**	35.29	35.79
	SSIM	0.4505	0.9493	0.6819	0.9500	0.9386	**0.9516**	0.9449	0.9491
Case 5	PSNR	20.23	33.82	25.56	32.18	32.95	**33.89**	33.19	33.84
	SSIM	0.4464	0.9176	0.6243	0.9114	0.9137	0.9259	0.9224	**0.9289**

[24] M. Mahmud and S. Ray, *Transfer learning using kolmogorov complexity: Basic theory and empirical evaluations*, Advances in Neural Information Processing Systems, 20 (2007).

[25] A. Maurer, *A vector-contraction inequality for rademacher complexities*, in International Conference on Algorithmic Learning Theory, Springer, 2016, pp. 3–17.

[26] A. Maurer, M. Pontil, and B. Romera-Paredes, *The benefit of multitask representation learning*, Journal of Machine Learning Research, 17 (2016), pp. 1–32.

[27] D. McNamara and M.-F. Balcan, *Risk bounds for transferring representations with and without fine-tuning*, in International Conference on Machine Learning, PMLR, 2017, pp. 2373–2381.

[28] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*, MIT press, 2018.

Table 16: Average test performance on DCmall by LRTV, E3DTV, LRTFDFR and their corresponding DLW-models. The best results in each **row** are in **bold**, and the second best results in each **row** are with underline.

	index	*Ori	N	T	TS	N+T	N+TS	T+TS	N+T+TS
target model 1 : **LRTV**									
Case 1	PSNR	32.63	31.17	28.41	29.24	**32.88**	30.44	32.86	32.60
	SSIM	**0.9194**	0.8782	0.8484	0.8402	0.9189	0.8610	0.9154	0.9046
Case 2	PSNR	**33.22**	31.89	28.06	29.92	33.19	30.89	33.12	33.12
	SSIM	**0.9317**	0.8926	0.8122	0.8585	0.9254	0.8730	0.9228	0.9169
Case 3	PSNR	31.86	31.28	28.09	28.88	**32.56**	30.38	32.15	32.23
	SSIM	0.9093	0.8870	0.8124	0.8377	**0.9193**	0.8672	0.9125	0.9068
Case 4	PSNR	33.86	33.80	29.64	31.60	**34.71**	32.85	34.46	34.59
	SSIM	0.9398	0.9265	0.8492	0.8970	**0.9452**	0.9147	0.9410	0.9385
Case 5	PSNR	31.89	31.19	28.98	28.78	**32.64**	30.47	32.31	32.16
	SSIM	0.9130	0.8846	0.8605	0.8303	**0.9196**	0.8656	0.9148	0.9049
target model 2 : **E3DTV**									
Case 1	PSNR	32.94	33.29	26.36	31.53	33.08	33.12	33.48	**33.74**
	SSIM	0.9115	0.9423	0.7751	0.9103	**0.9425**	0.9335	0.9385	0.9417
Case 2	PSNR	34.21	**34.49**	26.10	32.73	33.08	33.89	33.65	34.21
	SSIM	0.9293	**0.9551**	0.7441	0.9316	0.9437	0.9452	0.9425	0.9497
Case 3	PSNR	32.97	**34.21**	26.26	31.55	33.01	33.43	32.96	33.68
	SSIM	0.9138	**0.9501**	0.7452	0.9145	0.9413	0.9383	0.9355	0.9443
Case 4	PSNR	32.09	32.46	25.66	31.18	31.83	31.71	32.65	32.70
	SSIM	0.8961	0.9251	0.7378	0.8948	0.9204	0.9040	0.9217	0.9224
Case 5	PSNR	30.82	32.08	25.96	29.98	31.96	31.64	32.06	**32.50**
	SSIM	0.8822	0.9229	0.7660	0.8780	0.9240	0.9057	0.9194	**0.9244**
target model 3 : **LRTFDFR**									
Case 1	PSNR	33.31	33.19	26.13	31.66	32.68	32.86	33.22	**33.67**
	SSIM	0.9238	0.9323	0.7636	0.9081	0.9348	0.9257	0.9364	**0.9399**
Case 2	PSNR	**35.20**	34.21	25.76	32.54	32.67	33.51	33.23	33.87
	SSIM	**0.9516**	0.9435	0.7266	0.9229	0.9338	0.9354	0.9357	0.9434
Case 3	PSNR	33.44	**33.55**	25.90	31.47	32.42	33.00	32.56	33.28
	SSIM	0.9333	**0.9364**	0.7261	0.9041	0.9283	0.9281	0.9272	0.9358
Case 4	PSNR	36.37	**36.54**	27.24	35.13	33.94	36.12	34.46	35.44
	SSIM	0.9625	**0.9675**	0.7692	0.9546	0.9513	0.9642	0.9509	0.9608
Case 5	PSNR	19.82	33.68	26.71	31.65	32.84	33.27	33.13	**33.73**
	SSIM	0.2832	0.9406	0.7758	0.9109	0.9371	0.9330	0.9379	**0.9423**

[29] K. Muandet, D. Balduzzi, and B. Schölkopf, *Domain generalization via invariant feature representation*, in International Conference on Machine Learning, PMLR, 2013, pp. 10–18.

[30] N. Parikh and S. Boyd, *Proximal algorithms*, Foundations and Trends® in Optimization, 1 (2014), pp. 127–239.

[31] J. Peng, Y. Wang, H. Zhang, J. Wang, and D. Meng, *Exact decomposition of joint low rankness and local smoothness plus sparse matrices*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2022), pp. 1–16.

[32] J. Peng, Q. Xie, Q. Zhao, Y. Wang, L. Yee, and D. Meng, *Enhanced 3dtv regularization and its applications on hsi denoising and compressed sensing*, IEEE Transactions on Image Processing, 29

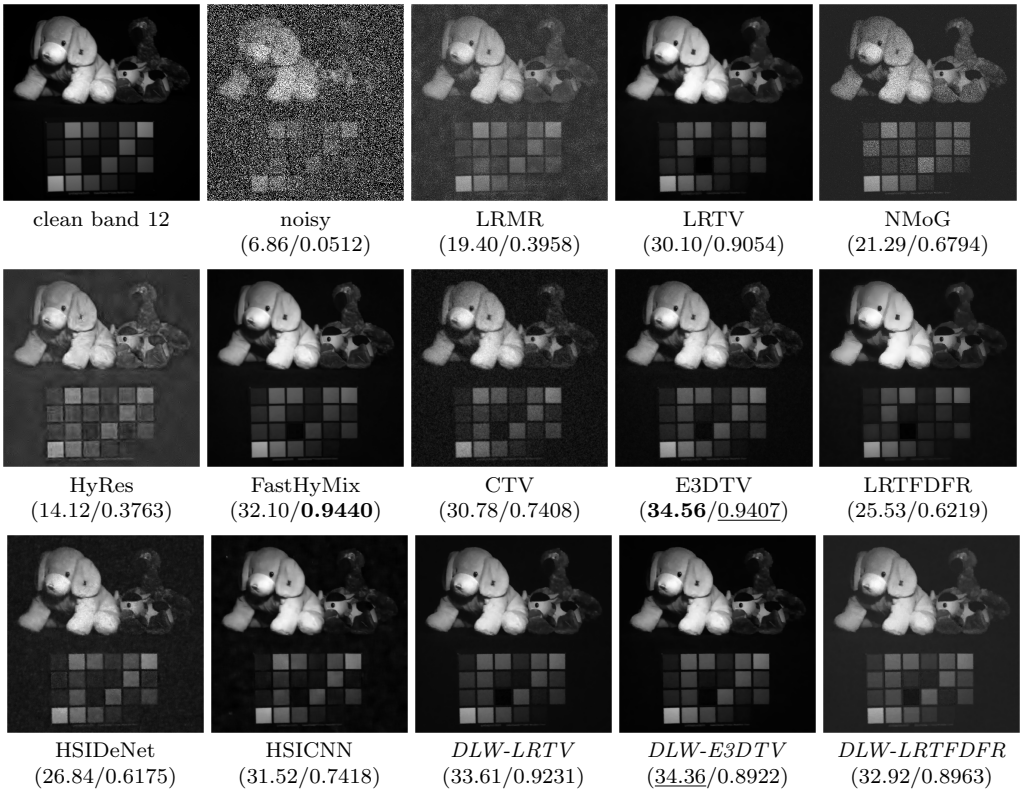


Figure 8: Visual comparison of the denoising performance on CAVE dataset obtained by different competing denoising methods

(2020), pp. 7889–7903.

[33] B. Rasti, M. O. Ulfarsson, and P. Ghamisi, *Automatic hyperspectral image restoration using sparse and low-rank modeling*, IEEE Geoscience and Remote Sensing Letters, 14 (2017), pp. 2335–2339.

[34] W. Shi, Q. Ling, G. Wu, and W. Yin, *A proximal gradient algorithm for decentralized composite optimization*, IEEE Transactions on Signal Processing, 63 (2015), pp. 6013–6023.

[35] J. Shu, D. Meng, and Z. Xu, *Learning an explicit hyperparameter prediction policy conditioned on tasks*, arXiv preprint arXiv:2107.02378, (2021).

[36] J. Shu, D. Meng, and Z. Xu, *Learning an explicit hyper-parameter prediction function conditioned on tasks*, Journal of machine learning research, 24 (2023), pp. 1–74.

[37] A. Sicilia, X. Zhao, and S. J. Hwang, *Domain adversarial neural networks for domain generalization: When it works and how to improve*, arXiv preprint arXiv:2102.03924, (2021).

[38] Y. Sun, Y. Yang, Q. Liu, J. Chen, X.-T. Yuan, and G. Guo, *Learning non-locally regularized compressed sensing network with half-quadratic splitting*, IEEE Transactions on Multimedia, 22 (2020), pp. 3236–3248.

[39] N. Tripuraneni, M. Jordan, and C. Jin, *On the theory of transfer learning: The importance of task diversity*, Advances in Neural Information Processing Systems, 33 (2020), pp. 7852–7862.

[40] D. Ulyanov, A. Vedaldi, and V. Lempitsky, *Deep image prior*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9446–9454.

[41] C. Wang, Z. Yan, W. Pedrycz, M. Zhou, and Z. Li, *A weighted fidelity and regularization-based method for mixed or unknown noise removal from images on graphs*, IEEE Transactions on Image Processing, 29 (2020), pp. 5229–5243.

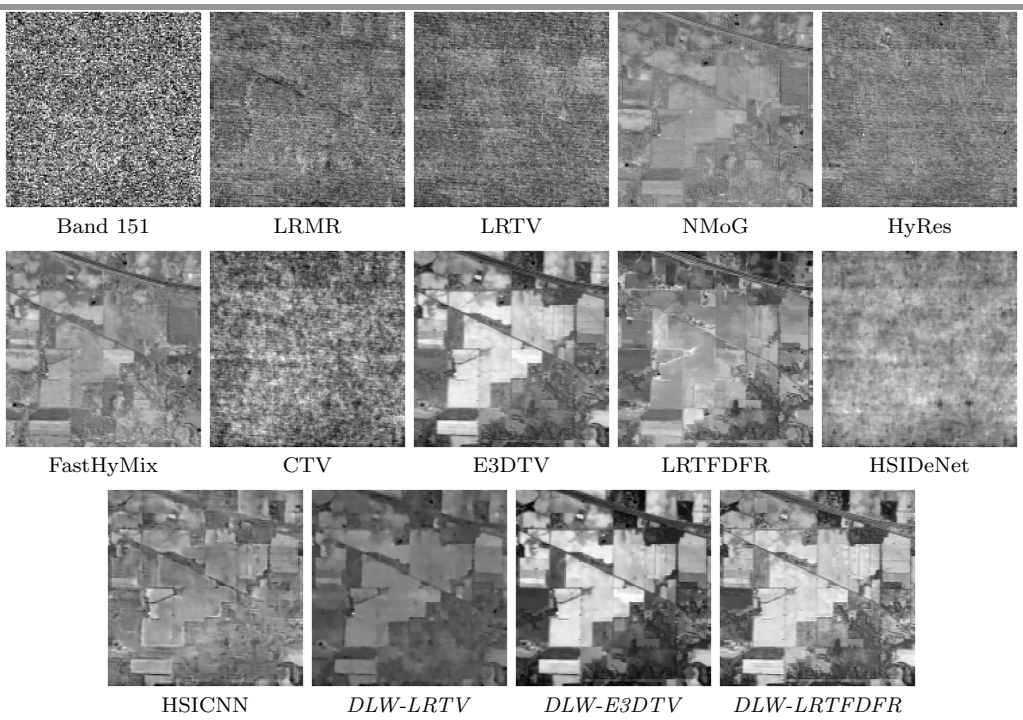


Figure 9: Visual comparison of the denoising performance on Indian Pine dataset obtained by different competing denoising methods

[42] X. Xie, Q. Wang, Z. Ling, X. Li, G. Liu, and Z. Lin, *Optimization induced equilibrium networks: An explicit optimization perspective for understanding equilibrium models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2022), pp. 1–14.

[43] J. Xu, L. Zhang, D. Zhang, and X. Feng, *Multi-channel weighted nuclear norm minimization for real color image denoising*, in Proceedings of the IEEE international conference on computer vision, 2017, pp. 1096–1104.

[44] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar, *Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum*, IEEE transactions on image processing, 19 (2010), pp. 2241–2253.

[45] Q. Yuan, Q. Zhang, J. Li, H. Shen, and L. Zhang, *Hyperspectral image denoising employing a spatial–spectral deep residual convolutional neural network*, IEEE Transactions on Geoscience and Remote Sensing, 57 (2018), pp. 1205–1218.

[46] Z. Yue, D. Meng, Y. Sun, and Q. Zhao, *Hyperspectral image restoration under complex multi-band noises*, Remote Sensing, 10 (2018), p. 1631.

[47] H. Zhang, W. He, L. Zhang, H. Shen, and Q. Yuan, *Hyperspectral image restoration using low-rank matrix recovery*, IEEE Transactions on Geoscience and Remote Sensing, 52 (2013), pp. 4729–4743.

[48] H. Zhang, L. Liu, W. He, and L. Zhang, *Hyperspectral image denoising with total variation regularization and nonlocal low-rank tensor decomposition*, IEEE Transactions on Geoscience and Remote Sensing, 58 (2019), pp. 3071–3084.

[49] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Y. Chen, and W. He, *Double-factor-regularized low-rank tensor factorization for mixed noise removal in hyperspectral image*, IEEE Transactions on Geoscience and Remote Sensing, 58 (2020), pp. 8450–8464.

[50] L. Zhuang and M. K. Ng, *Fasthymix: Fast and parameter-free hyperspectral image mixed noise removal*, IEEE Transactions on Neural Networks and Learning Systems, (2021), pp. 1–15.

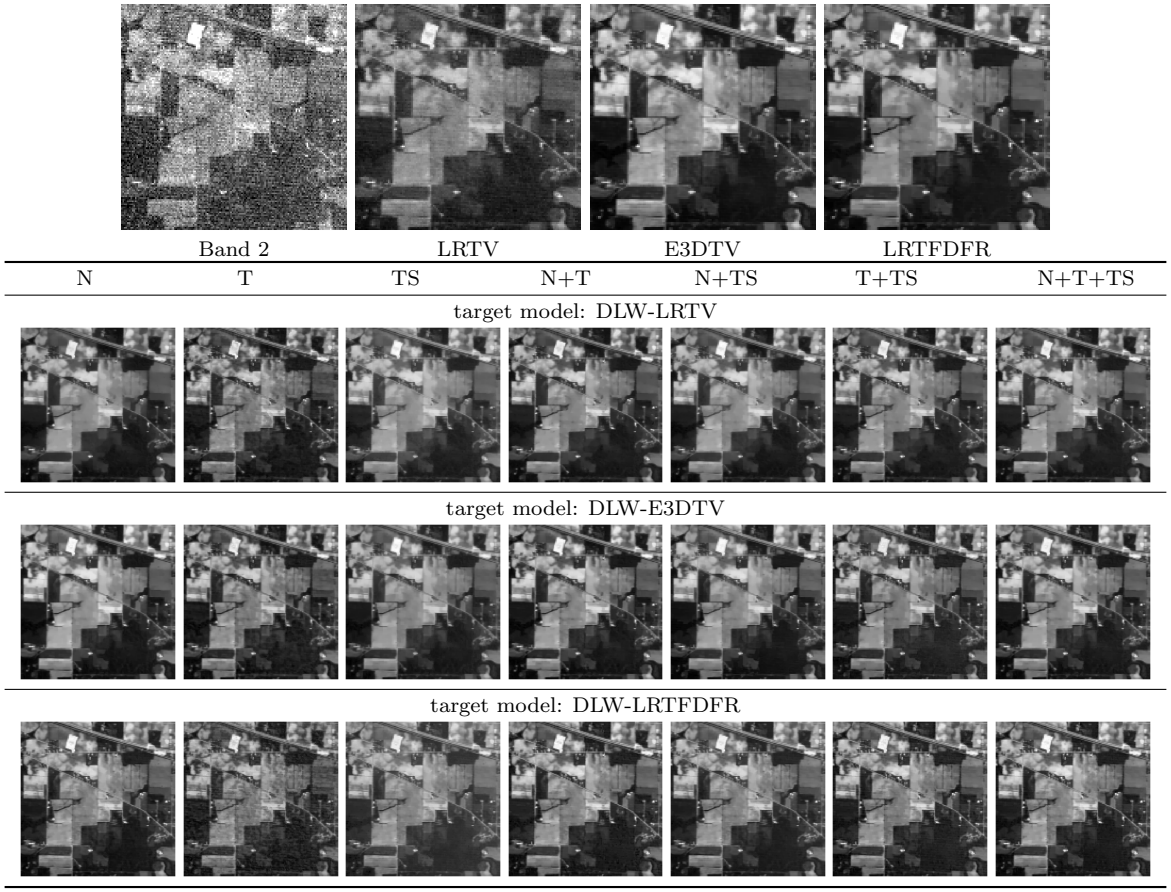


Figure 10: Visual comparison of the denoising performance on Indian Pine dataset obtained by LRTV, E3DTV, LRTFDFR and their corresponding DLW-models.

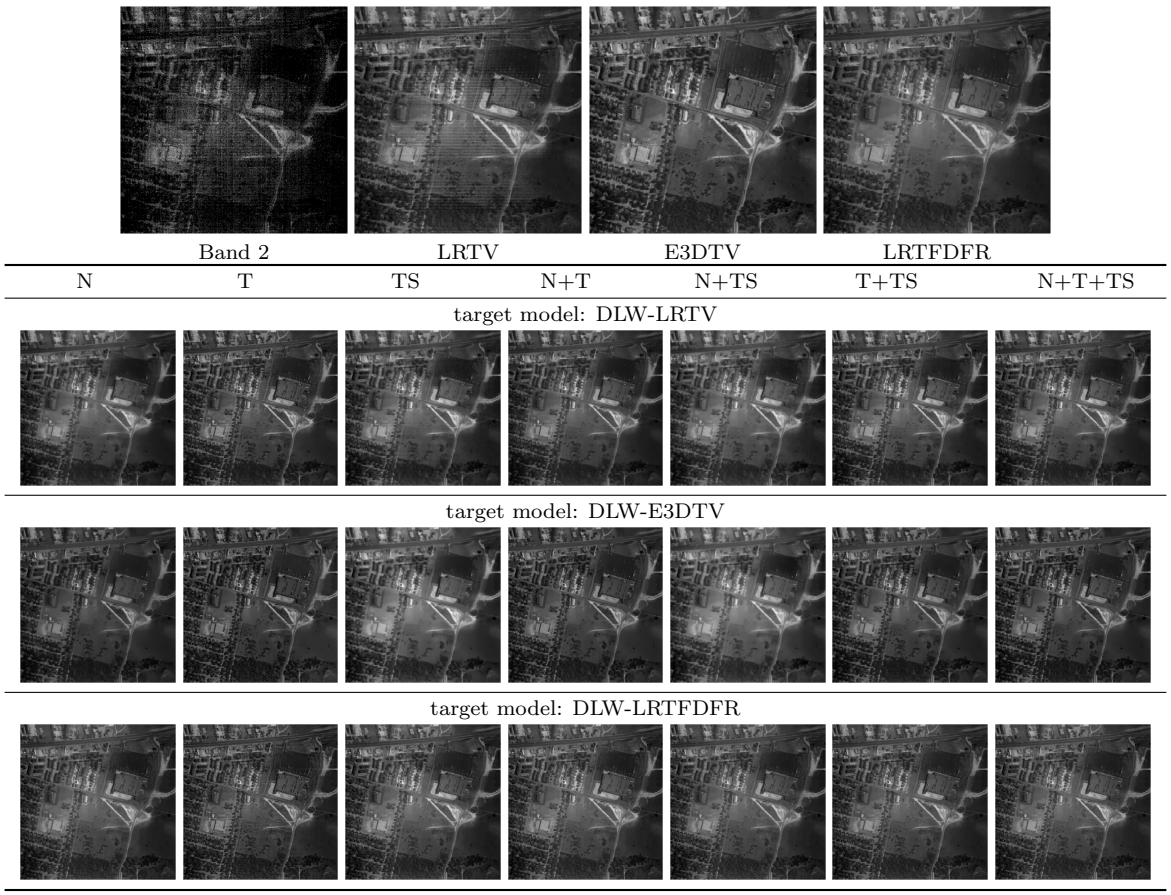


Figure 11: Visual comparison of the denoising performance on Urban dataset obtained by LRTV, E3DTV, LRTFDFR and their corresponding DLW-models.