Robust Representation Learning with Feedback for Single Image Deraining

Chenghao Chen and Hao Li

1. Department of Automation, Shanghai Jiao Tong University (SJTU), Shanghai, 200240, China.
2. École d’Ingénieurs SJTU-ParisTech (SPEIT), Shanghai, 200240, China.

\{1049164518, haoli\}@sjtu.edu.cn

Abstract

A deraining network can be interpreted as a conditional generator that aims at removing rain streaks from image. Most existing image deraining methods ignore model errors caused by uncertainty that reduces embedding quality. Unlike existing image deraining methods that embed low-quality features into the model directly, we replace low-quality features by latent high-quality features. The spirit of closed-loop feedback in the automatic control field is borrowed to obtain latent high-quality features. A new method for error detection and feature compensation is proposed to address model errors. Extensive experiments on benchmark datasets as well as specific real datasets demonstrate that the proposed method outperforms recent state-of-the-art methods. Code is available at: https://github.com/LI-Hao-SJTU/DerainRLNet

1. Introduction

Outdoor vision systems are used widely such as on intelligent vehicles and for surveillance. They sometimes suffer from rain pollution, which is undesirable in practice. To handle this problem, study on image deraining has appeared, which aims at removing rain streaks from image.

Some methods exploit specific a priori knowledge to clean images. For example, [8] introduces the dark channel; [7] maximizes the contrast among different target image regions. However, sparse rain streaks cannot be well removed by these methods. Some physical properties based deraining methods aim at separating the rain layer from the background layer via discriminative sparse coding [1, 20, 32], dictionary learning [14], and Gaussian mixture models [18]. However, specific a priori knowledge based methods are susceptible to complex, diverse, and changeable scenarios.

In recent years, deep learning based image deraining methods are rising. A deraining network can be interpreted as a conditional generator, and high-quality output images can be generated if conditional embedding features can characterize target image contents [24]. Most deep learning based deraining methods focus on designing novel network structures and guided features, such as residual based DDN [6], density based DID [33], recurrent structure based RESCAN [17], which can be regarded as strategies for embedding enhancement. Furthermore, [24] embeds mixed feature layers into the model and decode it into a clean image. Methods that do not consider uncertainty-induced model errors tend to loss details and incur halo artefacts in the generated image (see Fig. 1).

To handle uncertainty-caused model errors, we propose a robust representation learning with feedback for image deraining. Given an image with rain streaks, the objective is to output the residual map. Then the clean image can be obtained by subtracting the residual map from the rainy image as illustrated in Fig. 2. Since the embedding guides residual map generation, according to the image representation theory in [16], we try to find a functional relationship between basic embedding features and the optimal ones in the latent mapping space. The error detector and the feature compensator are designed to address model errors, for approximating the optimal embedding features. The contributions of the paper are three-fold:

1) We analyzed the impact of uncertainty in the training process and the possibility of model error remapping via convolutional neural networks.

Rainy image
DID [33]
Ours

Rainy image
RESCAN [17]
Ours

Figure 1. Image deraining. DID [33] tends to under-derain while RESCAN [17] tends to remove details of clean image.
2) Based on the idea of closed-loop control, the error detector and the feature compensator are designed for addressing model error via feedback mechanism.

3) Unlike existing image deraining methods that embed low-quality features into the model directly, we replace low-quality features with latent high-quality features. This is a new perspective for improving the deraining performance.

2. Related Work

2.1. Single Image Deraining

Single image deraining plays a basic role for deraining. Unlike video based methods which analyse the difference between adjacent images, it is hard to remove rain streaks from a single image for lack of temporal information and ill-posed nature. For the more challenging single image deraining, traditional methods based on specific optimisation methods [10, 27, 3, 32, 14] often tend to produce degraded images due to the limited mapping transformation. Recently, with the aid of the CNN, [6] first focused on high-frequency rainy information for rain streak removal and demonstrated impressive restoration performance of the negative rain streak mapping network. Motivated by the deep residual neural network [13], [5] proposes a deep detail network that is also trained in high-frequency domain to reduce range of intermediate variables from input to output. To handle heavy rain streaks, Li et al. decomposes a rainy image into a background layer and several rainy layers at different depth, and a recurrent CNN is proposed to remove rain streaks at state-wisely different depths [17]. Similarly, in [23], a context aggregation network combined with memory units is used to remove rain streaks stage by stage.

2.2. Representation Learning

A high-quality residual map can be generated if conditional embedding features are able to depict contents of the residual map. Yang et al. [30] decomposed a rainy image into a background layer and a rain streak layer, and located rain streaks via the binary map. However, both of removed details and remaining rain streaks on the clean image reflected the limitations of simple binary map guidance. Different from [30], Zhang et al. [33] took the rain density into account to guide the network learning. Compared with the inadequacy of the rain density label that only represents image-level rain features, Qian et al. [22] proposed to use the attention map to guide the residual map generation. We interpret that effectiveness mainly comes from the introduction of the rain streak features that guides the network to focus more on the rain streak regions. Later, Wei et al. [28] introduced that the combined multi-stream convolutional structure can better describe context information. Despite the improved performance, these methods learned the image-level or pixel-level representation without considering the detail loss caused by the uncertainty during training. Different from the structural improvement, a confidence guided method [31] studied the benefits from the residual map and its confidence. However, [31] simply used weight coefficients as confidence properties, which failed to consider the suitable confidence representation, error distribution complexity and error compensation. Thus, this method tends to miss details or introduce halo artifacts.

3. Feedback based Representation Learning

3.1. Problem Formulation

In [15, 4], the authors explained two types of uncertainty i.e. epistemic uncertainty a.k.a model uncertainty or systematic uncertainty, and aleatoric uncertainty a.k.a. statistical uncertainty that captures noise inherent in the observations. The variational inference can formulate epistemic uncertainty to compute variance. Maximum-aposterior or maximum-likelihood inference can formulate aleatoric uncertainty. To concisely describe our conditional optimization task, we model two output (conditional output and target output) and assume that probability distribution of each satisfies the Gaussian distribution. The minimisation objective, \( \mathcal{P} = - \log p(y_1, y_2 | f^W(x)) \), is given as:

\[
\mathcal{P} = - \log p(y_1 | f^{W_1}(x, f^{W_2}(x))) − \log p(y_2 | f^{W_2}(x))
\]

\[
\propto \frac{1}{2\sigma_1^2} ||y_1 - f^{W_1}(x, f^{W_2}(x))||^2
\]

\[
+ \frac{1}{2\sigma_2^2} ||y_2 - f^{W_2}(x)||^2 + \log \sigma_1 \sigma_2.
\]

where \( p(\cdot) \) represents the probability function, \( f^{W_i}(\cdot)(i = 1 \text{ or } 2) \) is the function of the corresponding network, \( x \) is the input rainy image and \( y_i(i = 1 \text{ or } 2) \) is the output. We denote the mean of distribution \( p(y_i | f^{W_i}(\cdot))(i = 1 \text{ or } 2) \) as \( y_i \) and the variance as \( \sigma_i^2 \). The operations with subscript label 2 are introduced to generate the embedding residual map to depict the contents of the residual map truth. The operations with subscript label 1 are introduced to generate the final residual map with the aid of the embedding residual map, and the clean image can be obtained by subtracting the residual map from the rainy image. Due to the uncertainty during training, the effect of changing the various modules in the network is limited. Thus we improve the deraining performance based on finding a functional relationship between the basic embedding residual map and the optimal one in the latent mapping space.

**Feedback mechanism.** After model training, some embeddings with large errors often increase uncertainty to degrade the deraining performance [15]. To obtain high-quality features, the spirit of closed-loop feedback is incorporated into the CNN. In the automatic control system, the closed-loop control is capable of reducing the error through the integral
function and the feedback. To apply the idea of the closed-loop control, we treat the training datasets as discrete inputs. As shown in Fig. 2, after generating embedding residual map \( \phi_t(I_t) \), we introduce the error detector (ED) with embedding residual map \( \phi_t(I_t) \) and rainy image \( I_t \) as inputs to learn the error between embedding residual map \( \phi_t(I_t) \) and corresponding residual map truth \( R_t \). The resulting error map is used to compensate the embedding residual map for the better feature representation.

**Effect analysis.** The role of the error detector is reflected in the training process and the results. For the training process, the effect of error compensation on the embedding residual map enables the R-to-R module (connecting this embedding residual map and the final output residual map) to be learned more accurately. With the training convergence of the R-to-R module, the final output loss function have a smaller impact on the O-to-R module (connecting the input rainy image and this embedding residual map), so that O-to-R module learns mainly based on the embedding feature loss function and acts as a feature guidance module. The decoupled learning is helpful to obtain appealing results [36]. To this end, the model error induced by uncertainty can be reduced by error compensation. For the results, the rectified embedding residual map are always more conducive to generate better deraining results as shown in Fig. 5.

### 3.2. CNN-based feedback process

**Error distribution complexity.** In general, the variable errors caused by the uncertainty during training can hardly be remapped by CNN due to the complexity of the error distribution. By abstracting error reciprocals as points that fluctuate above and below the zero value, Fig. 3.2 shows that operation of taking the absolute value of the error reciprocals can reduce the complexity of the error reciprocals distribution. Furthermore, the upper limit further simplifies the complexity of the error reciprocals distribution.

---

**Figure 2.** Compact overview of the proposed robust representation learning network structure (RLNet).

**Figure 3.** Full overview of the proposed robust representation learning network structure (RLNet). The ReLU functions after Conv and GN blocks are hidden for brevity. \( \Box \) represents information flow with index \( x \), \( \oplus \) is error compensation, \( \odot \) is concatenation operation, \( \otimes \) is pixel-wise multiplication, \( \oplus \) is pixel-wise addition and \( \ominus \) is pixel-wise subtraction. Zoom in to see small blocks better.
Small errors (corresponding to the large error reciprocals) that have little effect on the results are truncated by the upper limit, which is conducive to remap the error map. **Error detector.** Specifically, we adopt the reciprocal of absolute errors multiplied by the threshold parameter $\theta_1$ as training value to train the error detector, and use the sigmoid function at the end of the error detector. In such a case, the upper limit of the error detector output is 1, and the error detector output is closer to 0 as long as the error is larger. For the embedding residual map and the error detector, the corresponding optimization problems are:

$$\min_{\phi_1(\cdot)} \| R_{t \times 0.5} - \phi_1(I_t) \|_1, \tag{2}$$

$$\min_{\varphi(\cdot)} \| R_{t \times 0.5} - \varphi(I_{t \times 0.5}, \phi_1(I_t)) \|_1, \tag{3}$$

where $I_t$ is the rainy image, $R_{t \times 0.5}$ is the rainy image at 0.5 scale size of $I_t$, $R_t$ represents the residual map truth, $R_{t \times 0.5}$ is the residual map truth at 0.5 scale size of $R_t$, $\phi_1(\cdot)$ is the function trained to map $I_t$ as the residual map $(\times 0.5)$, $\varphi(\cdot)$ is the function of the error detector, and $\theta_1$ represents the threshold parameter. Due to the sparseness of rain streaks, reducing the size of the residual map by half can well represent the original residual map (see Fig. 5(d)(f)), and can greatly reduce the amount of calculation. Note that the last part of $\varphi(\cdot)$ is the sigmoid function and small errors are truncated by the upper limit. In addition, it should be emphasized that $|R_{t \times 0.5} - \phi_1(I_t)|$ in the optimization problem (3) is fixed, and the operation of taking the absolute value of the error reciprocal is very important for reducing the error distribution complexity. The absolute error map (named err in Eq. 4) and the rectified embedding residual map is calculated by the following rectification process:

$$\text{err} = \frac{\theta_1}{\varphi(I_{t \times 0.5}, \phi_1(I_t))} - \theta_1, \tag{4}$$

$$\phi'_1(I_t) = \phi_1(I_t) - \text{err}(1 - 2\phi_1(I_t)). \tag{5}$$

where 1 is all-one matrix and $\phi'_1(I_t)$ represents the rectified embedding residual map. In general, absolute errors (see Eq. 4) cannot compensate features well. For this reason, we introduce a method to reasonably use absolute errors.

For the generated clean image, regions that should be rain-free always exist rain streaks due to heavy rain regions involved in the training process. Specifically, for the local optimal solution with locally similar inputs composed of rain and backgrounds, small pixel values and large pixel values of the residual map continuously fit the network to each of them during training, which indicates that smaller pixel values of the residual map are more likely to be generated by the network with larger values due to the pull of heavy rain pixels during training, especially when backgrounds of input rainy images are similar such that inputs

![Figure 4. Abstract mapping from feature points (X-axis) to error reciprocals (Y-axis).](image)

(a) Distribution of 99 random error reciprocals within a certain range. (b) Distribution of random error reciprocals after taking the absolute value and upper limit truncation.

for the network convolution are similar. Hence, the embedding residual map $\phi_1(I_t), (0 \leq \phi_1(I_t) \leq 1)$ can describe the trend of corresponding errors. Note that the magnitude of embedding residual map values is much larger than error values. We multiply the absolute error map by the coefficient $(1 - 2\phi_1(I_t))$ that represents the confidence map of the absolute error map. To this end, the transformed error map (see Fig. 5(e)) are used to compensate the embedding residual map as shown in Eq. 5. The embedding residual map value from small to large corresponds to the error map value from negative to positive in terms of probability.

![Figure 5.](image)

(a) Rainy image. (b) Drained result of RLNet. (c) Groundtruth. (d) Embedding residual map $(\times 0.5)$ not rectified by the error detector. (e) Error map multiplied by 10 for visualization. (f) Final generated residual map.

**Feature compensator.** Similarly, our goal in the feature compensator is to design a light-weight module to obtain the high-quality embedding residual map to guide other embedding residual map. In this case, the embedding residual map that needs to be rectified by the error detector becomes more accurate to reduce the difficulty of error remapping. According to Eq. 1, we wish to obtain a residual embedding map less affected by uncertainty. Recalling that the CNN is of the complex nonlinear function, we adopt the CNN-based module (see Fig. 3(d)) to transform residual map truth $R_t$ for error compensation, which can be learned as follows:

$$\min_{\phi_1(\cdot), \omega_1(\cdot)} \| R_{t \times i} + \theta_2\omega_1(R_{t \times i}) \|_1 + \zeta, \tag{6}$$

where $\zeta = P(\omega_1(R_{t \times i})) = \lambda \times \| \omega_1(R_{t \times i}) \|^2_2$, $P(\cdot)$ means the regularizer, $\lambda$ is the setup parameter, $R_{t \times i}$ represents the
residual map at \( i \) (\( i = 0.25 \) or \( 0.5 \)) scale size of \( R_t \). \( \phi_i(\cdot) \) is the function trained to map \( I_i \) as the transformed residual map at \( i \) scale size (as shown in Fig. 6(d)(e)), \( \omega_i(\cdot) \) represents the transformation function and \( \theta_2 \) is the transformation parameter that determines the level of transformation. As \( \omega_i(\cdot) \) constrains the error by transforming features, the loss of details caused by uncertainty will be reduced.

![Image](51x578 to 127x624)

Figure 6. (a) Rainy image. (b) Derained result of RLNet. (c) Groundtruth. (d) Embedding residual map (×0.25) output by the feature compensator. (e) Embedding residual map (×0.5) output by the feature compensator. (f) Final generated residual map.

### 3.3. Network Architecture

**Feature Fusion Residual Block.** We adopt the group normalization [29] and the SE block [17] for the better feature fusion. As shown in Fig 3(b), the feature fusion residual block (FFRB) is formulated as:

\[
FFRB(x) = SE(GN(Conv(Res(x))))
\]

where \( x \) is the input signal and \( Res(\cdot) \) represents residual block containing group normalization (GN). Since the better performance of group normalization than batch normalization and instance normalization when the batch size is small [29], the group normalization is used to reduce internal covariate shift. By using SE block [17], the feature channels with more contextual information will be intensified by the larger weight coefficient offered by SE block. Mathematically, SE block can be expressed as:

\[
S(x) = \text{Sigmoid}(FC(ReLU(FC(GAP(x)))) \odot x),
\]

where \( GAP(\cdot) \) is the global average pooling and \( FC(\cdot) \) is the fully connected layer.

**Multi-stream Residual Architecture.** The proposed de-raining network is built upon the encoder-decoder architecture that is widely adopted in image de-raining [24, 31]. A large receptive field induced by encoder-decoder module U-FFRB is able to obtain context information. Since rain streaks commonly at different shape, density and scale, only utilizing one-stream CNN [23, 24, 31] may lose some useful information. Motivated by the effectiveness of multi-scale convolutional kernels [33, 28], we use the multi-stream residual network to extract multi-scale concentrated features. The kernel sizes of the FFRB belonging to their respective streams are set to 3, 5, and 7, respectively, to find back the lost details. From Fig 3(a), the multi-stream architecture can be described as:

\[
M(x) = [U_{3 \times 3}(x), U_{5 \times 5}(x), U_{7 \times 7}(x)],
\]

where \( U_{i \times j}(\cdot) \) represents U-FFRB module with the kernel size \( i \), and \( [\cdot] \) is the concatenation operation.

**Error detector and feature compensator.** For the feature compensator, an additional feature transformation module is introduced to adaptively transform residual map truth, and a concise encoder-decoder branch is constructed as shown in Fig 3(d) to learn the transformed residual map. For the correlation between different streams, the learned transformed residual map is copied into three copies and respectively embedded into three streams with skip-connection as shown in Fig 3(a). For the error detector, two concise branches containing FFRB are constructed to learn the residual map and the error map. Noted that one of the branches constructed for error map generation contains the encoder-decoder module. The error map is used to compensate corresponding embedding residual map (see Fig 3(a)).

**Refinement module.** The main goal of the refinement module is to finely adjust the feature maps clustered together. We use the spatial pyramid pooling [9] to further obtain multi-scale features. The scale factors are set to 4, 8, 16 and 32, respectively. For the feature maps with different sizes, the point-wise convolution is utilized to reduce their channels and the up-sample operation adopting the nearest interpolation is utilized to restore original size. As shown in Fig 3(a), the down-up structure can be formulated as:

\[
D(x) = [(Conv(x \downarrow 4)) \uparrow 4, (Conv(x \downarrow 8)) \uparrow 8, (Conv(x \downarrow 16)) \uparrow 16, (Conv(x \downarrow 32)) \uparrow 32, x].
\]

The next seven resblocks with the group normalization [29] are designed as shown in Fig 3(c).

### 3.4. Loss Function

The squared penalty of Mean Square Error (MSE) penalizes large errors and tolerates small errors, which tends to produce the over-smoothed image. Thus, Mean Absolute Error (MAE) is used to obtain better robustness. For the error detector rectifying the embedding residual map, we minimize the following two loss functions:

\[
L_{e1} = \|R_{t \times 0.5} - \phi_1 (I_t)\|_1,
\]

\[
L_{e2} = \|\frac{\theta_1}{\|R_{t \times 0.5} - \phi_1 (I_t)\|} - \varphi (I_t \times 0.5, \phi_1 (I_t))\|_1,
\]

Note that \( |R_{t \times 0.5} - \phi_1 (I_t)| \) in Eq. 12 is fixed. For the feature compensator, the loss function and the regularizer
are formulated as follow:

\[
L_c = \sum_{i \in (0, 25, 0.5)} \| R_{t \times i} + \theta_2 \omega_i(R_{t \times i})R_{t \times i} - \phi_i(I_t) \|_1, \tag{13}
\]

\[
L_p = \| \omega_i(R_{t \times i}) \|_2^2, \tag{14}
\]

In the proposed model, the final output residual map is refined via using the following loss function:

\[
L_f = \| R_t - f(I_t) \|_1, \tag{15}
\]

where \( f(\cdot) \) represents the overall network function. Furthermore, motivated by the goal of generating derained image to approximate its ground-truth image, we utilize the SSIM function [37] as the additional evaluation metric for the generated clean image. It is formulated as follows:

\[
L_{SSIM} = -SSIM(B_t, I_t - f(I_t)), \tag{16}
\]

where \( B_t \) is the clean image truth. The combination of the MAE based loss function and SSIM loss function can preserve the per-pixel similarity as well as preserving the global structure. The overall loss function used to train proposed RLNet is formulated as follows:

\[
L_{all} = L_f + L_{SSIM} + \lambda L_p + \lambda_1 L_e + \lambda_2 L_e + \lambda_3 L_c, \tag{17}
\]

where \( \lambda, \lambda_1, \lambda_2 \) and \( \lambda_3 \) are setup parameters.

4. Experiments and Results

4.1. Experiment Settings

Datasets. For fairness, we use the same benchmark datasets as other methods. DID-data with different rain magnitudes provided by Zhang et al. [33] contains 12000 training images and 1200 testing images, Rain800 collected by Zhang et al. [34] contains 700 training images and 100 testing images. Two datasets are synthesized by Yang et al. [30], namely Rain200H and Rain100L. Rain200H (with heavy rain streaks) consists of 1800 training images and 200 testing images, and Rain100L (with light rain streaks) consists of 200 training images and 100 testing images. In addition, [23, 26] supply some real-world rainy images as a test set. Deraining methods are trained on the synthetic dataset Rain200H but are tested on real-world images.

Training Details. During training, a image pair (or its horizontal flip) is resized as 512 × 512 for training. We implemented all experiments on a Titan Xp GPU and use a batch size of 1. The first stage is used to train the error detector and the feature compensator with the initial value \( \lambda = 0.01, \lambda_1 = 0.6, \lambda_2 = 0, \lambda_3 = 0.6, \theta_1 = 0, \theta_2 = 0.05 \) Adam is used as the optimizer to train the model and ends after 90 epochs. The initial learning rate is 0.0002 and divided by 5 when reaching 50, 65, 80 epochs. When reaching 20 epochs, \( \theta_2 \) is set as 0.15. When reaching 30 epochs, \( \lambda_2 \) is set as 6. By using the pre-trained weights obtained in the first stage, the fine-turning training process with the initial value \( \lambda = 0.01, \lambda_1 = 0.6, \lambda_2 = 0, \lambda_3 = 0.6, \theta_1 = 0.15, \theta_2 = 0.05 \) is as follows: The initial learning rate is 0.0002 and divided by 2 every 30 epochs. The 240 epochs in total. When reaching 30 \( \times K \) \((K = 1, 2, 3, 4, 5, 6)\) epochs, \( \lambda_2 \) is set as 0. When reaching 30 \( \times K + 15(K = 0, 1, 2, 3, 4, 5)\) epochs, \( \lambda_3 \) is set as 0.6.

Quality Comparisons. Peak Signal to Noise Ratio (PSNR) [11] and Structural Similarity Index (SSIM) [37] are adopted to assess the performance of different methods on benchmark datasets. Since there is no ground-truth image in real-world datasets, NIQE [21] is adopted to evaluate the generalization ability of different methods on real datasets.

4.2. Ablation Study

We conduct all ablation experiments on Rain200H for its heavy rain streaks distributed heterogeneously.

Absolute value operation. Using the non-absolute error map to directly add to the embedding residual map, a network termed as RLNet- with simple error compensation is constructed. Since the error map as a variable target is difficult to fit, even if we double the parameters of the error detector for RLNet-, the resulting 27.70 dB only exceeding the original size model RLNet- in Table 2 by 0.13 dB.

Threshold parameter \( \theta_1 \). In order to obtain a light-weight error detector, the absolute value operation is adopted and the threshold parameter \( \theta_1 \) is introduced to dynamically adjust upper limit of error reciprocals for better error map remapping as shown in Eq. 12. We keep a certain \( \theta_2 = 0.15 \)
Table 1. PSNR and SSIM comparisons on four benchmark datasets. Red and blue colors are used to indicate $1^{st}$ and $2^{nd}$ rank, respectively. $b$ denotes some metrics of this method are copied from [19]. $c$ denotes the model is re-trained due to no pre-trained weight provided by the author. To be fair, UMRL [31] and MSPFN [12] are fine-tuned with the Rain100L training set when evaluated on the Rain100L test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>Rain200H</th>
<th>Rain800</th>
<th>Rain100L</th>
<th>Rain700L</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>29.42/0.88</td>
<td>23.50/0.94</td>
<td>30.00/0.92</td>
<td>23.80/0.96</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.89/0.96</td>
<td>0.89/0.98</td>
<td>0.89/0.98</td>
<td>0.89/0.98</td>
</tr>
</tbody>
</table>

Table 2. Effect of threshold parameter $\theta_1$.

<table>
<thead>
<tr>
<th>Method</th>
<th>RLNet</th>
<th>RLNet−E</th>
<th>RLNet±E</th>
<th>E-Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>27.75</td>
<td>27.51</td>
<td>28.05</td>
<td>24.45</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.876</td>
<td>0.851</td>
<td>0.880</td>
<td>0.791</td>
</tr>
</tbody>
</table>

Table 3. Effect of the error detector. $-E$ means that RLNet does not contain the error detector. $\pm E$ denotes that RLNet uses the error detector in training but removes it after training. RLNet is our default model containing the error detector.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMRL</td>
<td>23.31</td>
<td>0.851</td>
</tr>
<tr>
<td>MPEN</td>
<td>27.51</td>
<td>0.880</td>
</tr>
<tr>
<td>MSPFN</td>
<td>24.45</td>
<td>0.791</td>
</tr>
</tbody>
</table>

Table 4. Effect of transformation parameter $\theta_2$.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Network</td>
<td>28.74</td>
<td>0.889</td>
</tr>
<tr>
<td>FFBR</td>
<td>28.81</td>
<td>0.895</td>
</tr>
<tr>
<td>Multi-stream Embedding</td>
<td>28.87</td>
<td>0.893</td>
</tr>
<tr>
<td>E-Detector</td>
<td>28.87</td>
<td>0.895</td>
</tr>
</tbody>
</table>

Table 5. Ablation study on different modules.

<table>
<thead>
<tr>
<th>Model</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>25.91</td>
<td>27.13</td>
<td>27.42</td>
<td>27.39</td>
<td>27.40</td>
<td>28.69</td>
<td>28.87</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.830</td>
<td>0.833</td>
<td>0.847</td>
<td>0.846</td>
<td>0.847</td>
<td>0.881</td>
<td>0.895</td>
</tr>
</tbody>
</table>

fixed and change the $\theta_1$ to find a better parameter setting for the error detector. Table 2 lists the PSNR and SSIM values of four RLNet models with $\theta_1 = 0.03, 0.04, 0.05, 0.06$. When $\theta_1 = 0.03, 0.04, 0.06$, the RLNet performs a little inferior to RLNet with $\theta_1 = 0.05$. It can be interpreted as the $\theta_1$ that is too large will weaken the role of the error detector, and the $\theta_1$ that is too small will increase the difficulty of error map remapping. Then we set the better parameter setting ($\theta_1 = 0.05$) for the proposed RLNet.

The results in Table 3 verify the effect of the error detector in the training process and results. In addition, we apply the embedding residual map and the error detector to the base network UMRL$^+$ [31]. The obtained UMRL$^+$+E achieves 2.14 dB performance improvement.

Transformation Parameter $\theta_2$. We keep a certain $\theta_2 = 0.05$ fixed and change the $\theta_2$ to find a better parameter setting for the feature compensator. Table 4 lists the PSNR and SSIM values of four RLNet models with $\theta_2 = 0.05, 0.1, 0.15, 0.2$. RLNet with $\theta_2 = 0.05, 0.1, 0.2$ performs a little inferior to RLNet with $\theta_2 = 0.15$. For the too small value of $\theta_2$ weakening the role of the feature compensator and the too large value of $\theta_2$ increasing the complexity of feature distribution, it is resonable to see those results from Table 4. Hence, we set $\theta_2 = 0.15$ for the RLNet.

Network Architecture. As shown in the Table 5), M1 denotes a single-stream U-net (kernel size is 3) without other modules. The base network, M1 itself enables the PSNR and SSIM to reach 26.91 dB and 0.830% respectively. M2 replaces the residual block with FFBR for M1. The FFBR increases the PSNR by 0.22 dB and the SSIM by 0.5%. Moreover, M3 modifies single-stream mechanism of M2 to multi-stream mechanism, with which M3 surpasses M2 by 0.29 dB and 1.4%. When compared with M3, M4 with the unrectified embedding residual map hardly improves performance. M6 adds the error detector to M4. By comparing M6 and M4, it can be seen that error detector by itself contributes 1.29 dB and 3.4%. M5 in Table 5) is obtained from M6 by removing the error detection oriented loss function $L_{e}$ but keeping all other conditions unchanged. Experimental results show that M5 no longer achieves the same performance as M6 but has degraded performance as M4. This implies that the proposed error detection mechanism does play an important role in enhancing the model performance. Besides, the RLNet±E in Table 3 that uses the error detector in training but removes it after training will finally have the same model size as the RLNet without using error detector at all, yet the former outperforms the latter. M7 adds $L_{e}$ to M6 (i.e. $\theta_2$ is changed from 0 to 0.15). The results show that M7 with the feature compensation obtains the better performance over M6 by 0.18 dB and 1.4%. The LPIPS [35] values of M4 (without error detector), M5 (without $L_{e}$) and M6 (with error detector) are 0.053, 0.052 and 0.041 respectively when tested on Rain200H. From Fig. 7, one can see that the error detector and the feature compensator further improve deraining performance.

4.3. Experiments on Benchmark Datasets

Ten widely adopted methods are compared with our network, including two traditional methods, i.e., DSC [20] and LP [18], and eight state-of-the-art deep learning based methods, i.e., DDN [6], DID [33], RESCAN [17], SPANet [26], UMRL [31], PReNet [23], MSPFN [12] and RCDNet [25]. Results of our method and other methods are given in Table 1. We select seven images from four benchmark datasets to visually validate the qualitative performance of different methods. Specifically, visually one can see in the second and third columns of Fig. 10, though the light rain streaks can be removed by SPANet [26] and DId [33], remaining rain streaks with missing image de-
tails usually exist when the rain is heavy. By comparing fourth, sixth and seventh rows of Fig. 10, the UMRL [31], RESCAN [17] and MSPFN [12] are hard to detect heterogeneously distributed rain streaks in different regions and recover clean image details well. We also test the derained image by DeepLabv3+ [2] as shown in Fig. 8. In addition, we randomly select 400 image pairs (512 × 512) to evaluate average time and performance of different methods as shown in Table 6. The error detector that takes little running time brings the considerable performance improvement. The FLOPs (in × 10^11) of RESCAN, MSPFN, RCDNet, RLNet±E, and RLNet are 1.3, 24.2, 7.8, 1.6 and 2.0.

### 4.4. Experiments on Real Rainy Images

Using the real-world datasets [26] cropped to 481×321, we compare proposed RLNet with other methods as shown in Table 7. In addition, from Fig. 9, it can be observed that RLNet, UMRL [31] and RESCAN [17] outperform DID [33] and MSPFN [12] in removing heavy rain streaks from real rainy image. Specifically, as can be seen from the Fig. 9, there are more or less visible rain streaks in the results by DID [33] and MSPFN [12], while RLNet, UMRL [31] and RESCAN [17] can generate the clean image. For the fourth, fifth and sixth columns of the Fig. 9, the blurs and halo artifacts with missing image details exist in the results by UMRL [31] and RESCAN [17], while the rain streak removal results by RLNet are high-quality and clear.

### 5. Conclusion

We have proposed the RLNet for single image deraining. Based on the CNN in the presence of uncertainty, a new method for error detection and feature compensation is proposed for latent high-quality representation learning. An iterative optimization scheme that unrolls image optimization and error optimization with image priors and correction terms is presented. Experiments demonstrate that the proposed RLNet is robust enough to detect heterogeneous rain streaks and recover details for real rainy images. Taking the portability into account, the proposed method for error detection and feature compensation can be selectively incorporated into learning based image deraining networks.


