ABSTRACT Recent studies have shown that a super-resolution generative adversarial network (SRGAN) can significantly improve the quality of single-image super-resolution. However, existing SRGAN methods also have certain drawbacks, such as an insufficient feature utilization, a large number of parameters. To further enhance the visual quality, we thoroughly studied three key components of SRGAN, i.e., the network architecture, adversarial loss, and perceptual loss, and propose a DenseNet with Residual-in-Residual Bottleneck Block (RRBB), called a residual bottleneck dense network (RBDN), for single-image super-resolution. First, to improve the utilization of features between the various layers of the network, we adopted a dense cascading connection between layers. At the same time, to reduce the computational cost, we added a bottleneck structure to each layer, greatly reducing the number of network parameters and accelerating the convergence speed of the training process. Second, the proposed RRBB, as the basic network building unit, removes the batch normalization (BN) layer and employs the ELU function to reduce the opposite effects in the absence of BN. In addition, we applied an improved overall loss function during the model training process to stably train the model and further improve the realism of the reconstructed high-resolution image. To prove the superiority of our proposed model, we conducted a comprehensive and objective evaluation of the Peak Signal-to-Noise Ratio, structural similarity, learned perceptual image patch similarity, and other evaluation indicators obtained from the three test sets, i.e., Set5, Set14, and BSD100, from the recent state-of-the-art model. Finally, we conducted qualitative and quantitative analyses of the results obtained in terms of the evaluation indicators, the authenticity of the restored HR images, and textural details, which show the superiority of the RBDN model.

INDEX TERMS Residual-in-residual bottleneck block, residual bottleneck dense network, bottleneck, ResNet.

I. INTRODUCTION

Image super-resolution (SR) techniques reconstruct a higher-resolution (HR) image or sequence from the observed lower-resolution (LR) images. Usually the benchmarks are single-image super-resolution (SISR) tasks. This a well-known ambiguous problem because there are always multiple HR images corresponding to a single LR image. With the development of deep learning in recent years, many methods have been proposed to solve this problem. Dong et al. [1], [2] conducted the first successful attempt, proposing a three-layer Super-Resolution Convolutional Neural Network (SRCNN) which achieved a better performance than traditional methods. Kim et al. [3], [4] then improved the network depth of Super-Resolution using Very Deep Convolutional Networks (VDSR) and a deeply recursive convolution network (DRCN) for image super-resolution to 20 levels, and eased the training difficulty by introducing residual learning. Chen et al. [5]–[10] introduced an attention mechanism in the network and improved the original convolutional neural network. Because convolutional neural network (CNN)-based methods [13], [16], [18], [28]–[33] achieve an excellent performance based on indicators such as the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM), which can reflect the pros and cons of a super-resolution reconstruction, many different types of
CNN-based methods have been proposed. Among the many previously proposed solutions, most optimization schemes calculate the pixel distance between LR and HR images through the root mean square error. However, the PSNR tends to output smooth results without sufficient details because the metric is fundamentally different from a subjective evaluation based on human vision. Several perceptual driven methods have been proposed to improve the visual quality of the final restoration of realistic images, such as the introduction of perceptual loss [19], [20]. A variety of GAN-based networks that use residual blocks and pursue a higher visual quality have also been proposed, including a super-resolution using a generative adversarial network (SRGAN), a further enhanced super-resolution generative adversarial networks (ESRGAN), and a recurrent generative adversarial Networks (RGAN). Although GAN-based methods can produce the desired high fidelity of the output, owing to an overfill of the network parameters, the remaining characteristics of a low utilization rate, and some problems inherent to the network layer structure, a SR reconstruction still has areas for improvement.

The SRGAN method carries the SR reconstruction to a new level, and the SRGAN ensures that the PSNR and other indicators remain at a high level and higher-standard image texture details can be recovered. However, there is still a difference between the perception of an image recovered using an SRGAN and that of an image in the real world. Nevertheless, the appearance of an SRGAN has resulted in a new and valuable method for super-resolution reconstruction. In recent years, a variety of different methods based on the SRGAN network structure have emerged, and ESRGANs have been used to restore reconstructed images at an extremely a very high level [21]. The feature reuse in this type of network plays a significant role, and has provided the inspiration for our study. However, the introduction of feature multiplexing (the idea of DenseNet) [25] also has certain disadvantages, that is, increasing numbers of computations and network parameters.

In recent years, an increasing number of models have used a residual network (ResNet) as their own network architecture, and have shown a good generalized performance. For example, Ledig et al. [11] developed a super-resolution residual network (SRRResNet) using the idea behind ResNet. One of the reasons that make ResNet exceptionally popular is the simple design strategy, which introduces only one identity shortcut. An identity shortcut skips the residual blocks to preserve the features, which limits the representation power of the network [44], [45]. As a drawback of ResNet, it causes a collapsing domain problem, which reduces the learning capacity of the network [45], [46] proposed to mitigate such a problem using nonlinear shortcuts. Another simple but effective technique, i.e., a dense connectivity, was proposed in DenseNet [25] to facilitate the training of deep networks. DenseNet uses dense cascading for all subsequent layers to avoid a direct summation to maintain the features of the previous layer, which has been proven to achieve better results [47]. Nevertheless, DenseNet requires a large amount of GPU memory, which is more complex from an engineering perspective, and further increases the training time of DenseNet [47]. As the main reason why DenseNet requires more training time, it uses dense cascading in the network, which increases the computational load of the entire network. In short, neither ResNet nor DenseNet can achieve a better performance while consuming fewer GPU resources.

In this study, motivated by the recent advances in deep learning and an approach to overcoming the shortcomings of these network architectures, we propose a dense network with a residual-in-residual bottleneck block (RRBB), called an RBDN, for single-image super-resolution. In an RDBN, we not only introduce dense connections in the original network structure, we also introduce a bottleneck structure to reduce the huge amount of computing required in the network and accelerate the convergence speed. At the same time, inspired by Lim et al. [18] and Wang et al. [21], we deleted the BN layer in the original network structure, which can speed up the training and convergence speed and prevent problems such as a gradient disappearance and overfitting, and the BN layer might lead to artifacts in our final output results. To obtain the gain brought about by removing the BN layer from the entire network and reducing the negative effect caused by removing the BN layer, we replace the original ReLU function with the ELU function. The ELU function not only alleviates the disappearance of the mitigation gradient, it also accelerates the convergence speed. In an ELU function such as a BN layer, the average data can be closer to zero, and the computational complexity is lower than that of mass regularization. An ELU function has a negative component compared to an ReLU function, and is more robust to noise compared to an LReLU or PReLU function. Therefore, our study further improved the GAN-based image SR models through the improved method mentioned above.

To verify the effectiveness of our method, we applied the above ideas to our designed network structure and combined them with the antagonistic generation network in [25] to form an RBDN, thereby adapting to the real world and reaching a higher level of network structure, as shown in Fig. 1.

Overall, the main contributions of this paper as follows:

1) The current mainstream deep learning network architectures for an image super-resolution reconstruction include EDSR, SRGAN, and ESRGAN. These deep learning network architectures have certain disadvantages such as a low feature utilization, a large number of network parameters, and poor scalability.

2) To solve the above problems, we proposed our own model, i.e., a novel generator model equipped with the combination of a bottleneck and dense connection, which can maximize the information flow throughout lightweight networks. In this generator, we removed the BN layer to eliminate artifacts that the BN layer may bring to the final result. Simultaneously, to reduce
the negative impact on the network by removing the BN layer, we replaced the original ReLU function with an ELU function. In addition, we applied an improved overall loss function in the model training process to stably train the model and further improve the realism of the reconstructed high-resolution image. This model can recover images with a higher resolution at a faster convergence speed by using more specific eigenvalues.

3) We retrained a pre-trained VGG-19 model to extract the most valuable textural details and compute the perceptual loss, and applied a combination of perceptual loss, texture loss, total-variation loss, and content loss as the perceptual loss to achieve a wide-ranging effect while generating SR images.

4) During the pre-processing stage before the formal SR reconstruction, we obtained LR images and the corresponding Bicubic images through a pre-processing of the HR images. We then used RBDN, ESRGAN, SRGAN, and other models to achieve an SR reconstruction of the LR images generated during the first stage. We then applied a comprehensive combination of the performance metrics, including PSNR, SSIM, and learned perceptual image patch similarity (LPIPS), to monitor and assess the quality of the images.

II. RELATED WORK

SR reconstruction methods can be divided into two categories: traditional methods, such as sparse coding, and deep-learning-based approaches, which have become popular in recent years. Owing to the strong learning ability of deep learning methods, traditional learning methods are gradually being replaced by methods based on deep learning. Therefore, we focused on using deep neural networks to solve the SR reconstruction problem. The pioneer in this field was Dong et al. [1], [2], who proposed a shallow rerouted neural network (SRCNN) for SISR tasks to learn the mapping of LR-HR images in an end-to-end manner, and achieved a better performance than that in previous studies. A variety of network architectures later emerged in this area and have been used to this day, including residual blocks, densely connected networks [15], and residual dense networks [16]. Lim et al. [18], whose EDSR network model first applied the idea of removing the BN layer from the network structure. Ledig et al. [11] developed SRResNet for SR tasks using the concept of ResNet [12]. In addition, Zhang et al. [16] proposed residual dense blocks for SISR reconstruction and proposed a deep network of attention channel mechanisms, the PSNR of which reached a fairly high level. However, we should also note that, although the aforementioned model can obtain ideal PSNR indicators, in an actual application of image recovery, distortion indicators and perceived quality contradictions will occur, along with image edge sharpening, and other issues, causing the actual recovery of the image effect to be inconsistent with our expectations.

Today, perception-driven algorithms are mature and are used to improve the visual quality of the SR results. Based on the idea of proximity to perceived similarity [19], [20], the perceived loss is increased to enhance the visual effects by minimizing the perceived loss. Ledig et al. [11] proposed SRGAN, which achieves a multi-image output based on the perceived loss and counter loss. The manner in which SRGAN operates also facilitates what the next stage. Based on SRGAN, ESRGAN [21] introduces dense residual blocks (RRDBs) into the original structure and removes the BN layer [25], [26]. At the same time, the perceived loss is adjusted properly, and the advent of ESRGAN allows an SR reconstruction to reach a higher level.

In the development process of the SR reconstruction method based on deep learning, many network architectures appear to be similar to the deep network mentioned above. However, it should also be noted that although the above models can obtain ideal PSNR or SSIM indicators during the actual application process, the images restored by these approaches will have certain problems in terms
of the perceived quality and image edge sharpening. The image effect resulting in an actual recovery did not meet our expectations.

The advantages and disadvantages of SR algorithms are usually evaluated using several widely applied distortion-based metrics, such as the PSNR and SSIM. Combined with the above analysis, it is unreasonable to consider only the distortion index for a restored image. In recent years, Blau and Michaeli [43] have found that the distortion and perceived quality are contradictory, and thus it is more reasonable to consider the distortion index and perceived quality assessment in a comprehensive manner. In this study, we introduced the perceptual loss to make the generated image results more in line with human perception.

III. METHODOLOGY

In this section, to achieve the purpose of the present study, we elaborate on the design details of an RBDN, which will improve the overall perceptual quality of the SR when the distortion-based metrics (PSNR, SSIM) of the image are maintained at a reasonable level. In the following, we first describe our improved network architecture. We then describe the design method of the generator (the objective function of its source), and finally detail the loss function applied.

A. DETAILED DESCRIPTION OF THE FUNCTIONS AND NETWORK STRUCTURE USED IN OUR NETWORK

ReLU is the most commonly used activation function, which is simple in form and is highly nonlinear. The activation function used in SRGAN is ReLU. The ReLU function is defined as follows:

$$
\text{ReLU}(x) = \begin{cases} 
 x & x \geq 0 \\
 0 & \text{otherwise}
\end{cases}
$$

ReLU is highly nonlinear, which can speed up the convergence speed, alleviate the problem of a gradient disappearance and explosion, and simplify the calculation. However, ReLU changes all negative input numbers to zero, which can be extremely fragile during training. It easily causes a neuron inactivation, and the neurons will not be activated again at any data point. Where the ReLU domain is less than zero, the gradient is also zero, and thus the weight will not be adjusted during the descent process.

Leaky-ReLu is a variant of ReLU. Compared with ReLU, LReLU improves the piecewise function of a part whose domain is less than zero. We can set the $\alpha$ coefficient to ensure that LReLU has a corresponding output when the input is less than zero, which reduces the sparsity of ReLU. The activation function used by ESRGAN is LReLU, the definition of which is as follows:

$$
\text{LReLU}(x) = \begin{cases} 
 x & x \geq 0 \\
 \alpha x & \text{otherwise}
\end{cases}
$$

The LReLU function alleviates the problem in which the ReLU function may cause neuron inactivation. Although LReLU has a non-zero output in a domain of less than zero, it causes the nonlinearity to be less powerful than that of ReLU.

In our designed network structure, we used the ELU as the activation function. ELU, similar to batch regularization, can make the mean value of the data closer to zero, and the computational complexity of the ELU function will be lower than that of batch regularization. The definition of the ELU function is as follows:

$$
\text{ELU}(x, \alpha) = \begin{cases} 
 x & x \geq 0 \\
 \alpha (\exp(x) - 1) & \text{otherwise}
\end{cases}
$$

The linear part of the piecewise function allows the ELU to alleviate the disappearance of the gradient, and the nonlinear part allows the ELU to be more robust to changes in input or noise. The ELU function has a specific definition in the non-positive domain, which makes the output mean of the ELU close to zero to achieve a faster convergence rate.

Furthermore, our model introduces an auto-encoder composed of an encoder and a decoder. The encoder maps the input samples to the feature space, and the decoder maps the abstract features back to the original space to obtain the reconstructed samples. The encoder part compresses the data by reducing the number of neurons layer by layer to achieve a data dimensionality reduction and classification. The decoder part is based on the abstract representation of the data used to increase the number of neurons in a layer-by-layer manner, and finally realizes the reconstruction of the input sample.

B. NETWORK ARCHITECTURE

Our RBDN architecture mainly consists of four parts: an encoder, residual bottleneck dense network (RBDN) blocks, a decoder, and a clipping layer, as shown in Fig. 2. The encoder contains up-sampling and one convolutional layer, and the decoder contains one T-convolutional layer and one trainable project layer. We denote $I_{LR}$ and $I_{SR}$ as the input and output of the RBDN, respectively. The first convolutional layer extracts the features from the LR input as

$$
F_0 = H_{\text{Encoder}}(I_{LR}),
$$

where $H_{\text{Encoder}}(\cdot)$ denotes the up-sampling and convolution operations, and $F_0$ is the input to the basic RBDN block for further shallow feature extraction and global residual learning.

Supposing there are $M$ basic blocks in the RBDN, the output of the $m$-th RRBB ($m \in \mathbb{N}[1, M]$) can then be obtained by

$$
F_m = H_{\text{RRBB},m}(F_{m-1}) = H_{\text{RRBB},m}(H_{\text{RRBB},m-1}(\cdots (H_{\text{RRBB},1}(F_0)) \cdots ))
$$

where $H_{\text{RRBB},m}$ denotes the operation of the $m$-th RRBB, which can be a composite function of different operations, such as convolution and exponential linear units. Because $F_m$ is produced by the $m$-th RRBB by fully utilizing each convolutional layer within the block, we can view $F_m$ as a
local feature. More details about RRBB and RBDN are given in Sections III-B and III-C.

Now, the output of the RBDN blocks will pass through the decoder, which attempts to reconstruct the input data as

$$F_d = H_{\text{Decoder}}(F_m).$$

After extracting hierarchical features with a set of RRBB blocks, the estimated residual image after the decoder is subtracted from the LR input image, which can be represented as

$$F_{\text{DFF}} = H_{\text{DFF}}(F_{\text{Upsample}} - F_d) \quad (6)$$

where $F_{\text{DFF}}$ is the estimated residual image after the decoder is subtracted from the LR input image, and $F_{\text{Upsample}}$ is the result obtained after the up-sampling of the LR image.

Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and forces the pixel values of the reconstructed image to lie within the range of [0,255]. Reflection padding is also used before all Conv layers to ensure slowly varying changes at the boundaries of the input images, and we use a clipping layer in the SR space to obtain the RBDN output as follows:

$$I_{\text{SR}} = H_{\text{RBDN}}(I_{\text{LR}})$$

where $H_{\text{RBDN}}$ denotes the function of our RBDN.

C. RESIDUAL-IN-RESIDUAL BOTTLENECK BLOCK (RRBB)

To describe our proposed RRBB, we refer to the residual block (RB) in ResNet [12]. Compared to the original RB, we made the following improvements in the RRBB: We (1) delete the BN layer, (2) employ a dense connection between residual blocks, similar to the DenseNet structure [16], [18], (3) introduce a bottleneck structure to reduce the number of parameters and the number of calculations brought about by the dense connections, and (4) replace the ReLU activation function in the basic block with ELU. These improvements enable our RRBB to differ significantly from a residual block in SRGAN [11] and a dense block in ESRGAN [21], as shown in Fig. 3. The following sections describe these changes and how they allow for better results.

First, the presence of a BN layer has beneficial effects on the entire network, such as speeding up the training and convergence, avoiding gradient vanishing, and preventing an overfitting. However, the BN layer also has negative effects in that it reduces the absolute difference between data samples but highlights the relative difference. For instance, when the training and test datasets are quite different, the BN layer may reduce the absolute difference between the data samples and further limits the generalization ability of the network [21]. For this reason, the BN structure performs well in image classification and related tasks, but does not perform well in an SiSR task. Hence, deleting the BN is helpful in enhancing the model performance and reducing the computational complexity [21].

Next, to strengthen the feature propagation and reuse, we also use basic blocks similar to dense blocks [16], [16], [18] proved that the dense connection between different
convolutional layers can improve the performance in feature fusion. Although the introduction of dense connections will increase the calculation cost, we use the bottleneck structure to reduce the number of parameters and the number of calculations.

Finally, the ReLU function in a general RB is replaced with a better-performing ELU function, which improves the network performance while offsetting the negative effects of removing the BN layer. The ELU function can not only alleviate the problem of a gradient disappearance, such as ReLU and LReLU, but also robust to noise and has a faster convergence speed than LReLU. Combining all of the above ideas, we redesigned an improved basic block RRBB.

D. PROBLEM FORMULATION AND MINIMIZATION STRATEGY

In this section, we mathematically processed the design ideas proposed in section 3.2. In this framework, the objective function minimization problem of transforming an SR into a standard form is as follows:

$$\hat{x} = \arg\min_{\hat{x} \in \mathbb{R}^N} \frac{1}{2} \|y - H\hat{x}\|_2^2 + \lambda R_W(x) \triangleq \arg\min_{\hat{x} \in \mathbb{R}^N} F(\hat{x}),$$ (7)

where $\frac{1}{2} \|y - H\hat{x}\|_2^2$ is the data fidelity term used to measure the closeness of the solution to the target, and this term is associated with the image degradation model $y = Hx + n$ in which $y \in \mathbb{R}^{N \times s}$ and $\hat{x} \in \mathbb{R}^{N \times N}$ are the vectorized versions of the observed LR and HR images, respectively, $N \times N$ is the total number of pixels in an image, $s$ is the scaling factor associated with the down-sampling operator $H \in \mathbb{R}^{N \times s \times N \times N}$ that resizes the HR image $\hat{x}$, and the variable $n$ denotes the additive white Gaussian noise with a standard deviation of $\sigma$ (noise level). In addition, $R_W(x)$ is a regularization item related to the image prior [23], [24] and is defined as $R_W(x) = \sum_{k=1}^{K} \rho_k(L_k x)$, where $W$ is the network parameter, $\rho_k(\cdot)$ represents a potential function [25], and $L_k$ is a first-order or higher-order differential linear operator. In addition, $\lambda$ is the balancing factor for the data fidelity term and image prior.

In the minimization process, a proper optimization strategy is employed to find the optimal network parameters $W$ that minimize the objective function (7) to obtain the required latent HR image. Because the objective function may not be fully differentiable, it can be divided into smooth and non-smooth parts through the proximal gradient algorithm [26]. Therefore, the objective function (7) can be written in the following equivalent form:

$$\hat{x} = \arg\min_{\hat{x} \in \mathbb{R}^N} \frac{1}{2} \|y - H\hat{x}\|_2^2 + \lambda \sum_{k=1}^{K} \rho_k(L_k x) + \ell_{C(\epsilon)}(x),$$

where $\ell_{C(\epsilon)}(\cdot)$ is the indicator function on the convex set $C$, which can be computed using a trainable projection layer [24], which is defined as

$$\ell_{C(\epsilon)}(x) = \begin{cases} 0, & \text{if } \|y - x\|_2 \leq \epsilon \\ +\infty, & \text{otherwise} \end{cases}$$ (9)

where $\epsilon = \exp(\delta)\sigma \sqrt{N_t - 1}$ is the parameterized threshold with trainable parameter $\delta$ and the total number of pixels in the image $N_t$. Using the proximal gradient algorithm [26], the recursive solution of (8) is as follows:

$$x' = \text{prox}_{\eta'}\ell_{C}(x^{-1} - \eta' \nabla_x F(x^{-1})), \quad (10)$$
where $\eta'$ is the iteration step size, and $\text{prox}_{\eta'\ell_C}$ is the proximal operator of the gradient algorithm associated with the indicator function $\ell_C(x, \epsilon)$, which is defined as

$$P_C(z) = \arg\min_{x \in C} \|x - z\|_2^2 + \ell_C(x, \epsilon).$$

(11)

Based on a simple calculation, we obtain the gradient of $F(x)$ as

$$\nabla x F(x) = H^T(Hx - y) + \lambda \sum_{k=1}^{K} L_k^T \phi_k(L_kx),$$

where $\phi_k(\cdot)$ is the gradient of $\rho_k(\cdot)$, where $\phi_k$ corresponds to the ELU nonlinear activation function in our designed network. In Eqs. (7)–(11), the final solution of the objective function can be expressed as follows:

$$x_t = P_C \left( (1 - \eta'H^TH)x_t^{-1} + \eta'H^Ty \right. \left. - \delta \sum_{k=1}^{K} L_k^T \phi_k(L_kx_t^{-1}) \right).$$

(12)

where $\delta = \lambda \eta'$ denotes the trainable projection layer parameter.

In summary, we designed our generator network according to Eq. (12).

### E. NETWORK LOSSES

We employ four classes of loss functions to measure the reconstruction errors and train our generative adversarial network [27]. The overall loss function is formulated as follows:

$$L_{RBDN} = L_{\text{per}} + \alpha L_{\text{Gan}}^{Ra} + \beta L_{\text{tv}} + \gamma L_1$$

(13)

where $\alpha$, $\beta$, and $\gamma$ in Eq. (13) are the coefficients used to balance the different losses.

- **Perceptual loss ($L_{\text{per}}$):** This measures the semantic differences between images and helps the results produced by our trained network to achieve better perceptual effects, which is defined as follows:

$$L_{\text{per}} = \frac{1}{N} \sum_{i=1}^{N} \| \phi(G_{SR}(\hat{x}_i)) - \phi(y_i) \|_1,$$

(14)

where $\phi$ is the feature extracted from the pretrained network.

- **Texture loss ($L_{\text{Gan}}^{Ra}$):** This term focuses on the high frequencies of the output image. According to the framework of the generator based on the relativistic generative adversarial network [21], we define the relativistic average generator loss as follows:

$$L_{\text{Gan}}^{Ra} = -E_y [\log (1 - D_{Ra}(y, G_{SR}(\hat{x})))]
- E_{\hat{y}} [\log (D_{Ra}(G_{SR}(\hat{x}), y))].$$

(15)

where $E_y$ and $E_{\hat{y}}$ represent the operation of averaging all real $y$ and fake $\hat{y}$ data in the mini-batch, respectively.

- **Total-variation loss ($L_{\text{tv}}$):** This term mainly focuses on suppressing noise in generated images (25). In this study, it is defined as the sum of the absolute differences of the gradient discrepancy to produce sharpness in the
output image, as follows:

\[
L_{TV} = \frac{1}{N} \sum_{i=1}^{N} (\| \nabla_h G_{SR}(\hat{x}_i) - \nabla_h(y_i) \|_1 \\
+ \| \nabla_v G_{SR}(\hat{x}_i) - \nabla_v(y_i) \|_1),
\]

where \( \nabla_h \) and \( \nabla_v \) represent the horizontal and vertical gradients of the image, respectively.

- **Content loss \( (L_1) \):** This item is suitable for evaluating the 1-norm distance content loss between the generated result and the real image.

\[
L_1 = \frac{1}{N} \sum_{i=1}^{N} \| G_{SR}(\hat{x}_i) - y_i \|_1.
\]

### IV. RESULTS AND DISCUSSIONS

#### A. DATASETS AND EVALUATION METRICS

We mainly used DIV2K [34] as the training dataset, which is a dataset that contains 800 images with 2 K resolution and is mainly used for super-resolution reconstruction. The Flickr2K [37] and OutdoorSceneTraining (OST) [23] datasets were used to enrich the training set with more diverse textures. Previous research has found that richer texture information helps the generator produce more natural results. Our final training set consisted of 13,297 images, and the LR images we needed were obtained by subsampling the images from the above dataset. In the testing process, we used three benchmark test sets: Set5 [40], Set14 [41], and BSD100 [42].

In this study, we evaluated our training model using the PSNR, SSIM, and LPIP [35], as well as the training time. The PSNR and SSIM are distortion-based measures, whereas LPIPS is a perceptual metric based on human similarity judgments; thus, it is a subjective measurement. However, it is difficult to declare one metric to be absolutely better than the rest because each metric is tendentiously and partially used to evaluate the model in one dimension. A fair comparison is only possible by synthetically considering the subjective and objective metrics. In addition, we also considered the training time (TIME), which is a simple and efficient metric for evaluating all models with consistent parameters.

#### B. TRAINING DETAILS

The training process is divided into two phases. First, we train a model for the PSNR through the \( \ell_1 \) loss. The learning rate is initialized as \( 2 \times 10^{-4} \) and is decayed by 1/2 for every \( 2 \times 10^5 \) small-volume updates. We then use the trained PSNR-oriented model as our initial generator through the loss function (13). The super parameters \( \alpha \) and \( \gamma \) in Eqs. (13) are \( 5 \times 10^{-3} \) and \( 1 \times 10^{-2} \), respectively. Here, we set the initial learning rate to \( 1 \times 10^{-4} \), which is halved at \([125k, 250k, 500k, 750k]\) iterations, and we conducted a total of \( 1,000,000 \) iterations [49].

Throughout the training, we used the PGM method [26], and updated the generators and discriminator in the network alternately until convergence was reached. We ended up with an internal generator structure using a hierarchy and number of layers similar to those of ESRGAN [21] and a deeper model of 23 RRBBs, each of which contains 27 levels of functional layers. The above study was implemented using the PyTorch [39] framework and trained using the NVIDIA RTX8000.

In particular, the introduction of perceptual loss in Eq. (13) is also one of the details of our training: Pre-training with perceptual loss helps GAN-based methods to obtain more visually pleasing results because 1) it can avoid undesired local optima for the generator and 2) after pre-training, the discriminator receives relatively good super-resolved images instead of extreme fake ones (black or noisy images) at the very beginning, which helps it to focus more on a texture discrimination [21].

#### C. RESULTS AND ANALYSIS

1) **QUANTITATIVE COMPARISON**

We compared our methods with state-of-the-art perceptual-driven SRs for quantitative analysis methods including EDSR [18] and SRResNet [12], and with perceptual-driven...
TABLE 1. Quantitative comparison results of our model with other models on three test datasets. The best score in each line is highlighted in red, and the sub-optimal score is highlighted in blue. Because the datasets and network hyperparameters required for training and testing of different network structures vary widely, to ensure the comparability and fairness of the experimental results, we reproduced all models included in the table under the same experimental conditions, and re-obtained the indicators of each model. In particular, each indicator in the table is the average of the indicators obtained from the three experiments.

| Dataset | Metric | Bicubic | SRGAN | SRResNet | EDSR | ESRGAN | RBDN (ours) |
|---------|--------|---------|--------|----------|------|--------|-------------|
| Set5    | PSNR † | -       | 28.0289| 29.2508  | 29.1822| 28.9790| 27.8422     |
|         | SSIM † | -       | 0.8078 | 0.8642   | 0.8632| 0.7543 | 0.7882      |
|         | LPIPS  | -       | 0.0882 | 0.1354   | 0.1387| 0.0788 | 0.0748      |
| Set14   | PSNR † | -       | 24.7499| 26.2395  | 26.7959| 25.3925| 24.3409     |
|         | SSIM † | -       | 0.6127 | 0.7122   | 0.7448| 0.5796 | 0.6618      |
|         | LPIPS  | -       | 0.1663 | 0.2309   | 0.2349| 0.1329 | 0.1319      |
| BSD100  | PSNR † | -       | 24.1956| 26.2395  | 26.2265| 25.3265| 25.3409     |
|         | SSIM † | -       | 0.6245 | 0.7632   | 0.7114| 0.6592 | 0.6127      |
|         | LPIPS  | -       | 0.1980 | 0.3094   | 0.3146| 0.1614 | 0.1612      |

FIGURE 7. Visual comparison of our model with other state-of-art models on BSD100 test dataset at ×4 SR. The ‘PSNR/LPIPS’ metrics: (b) -/-, (c) 28.5119/0.2537, (d) 28.5198/0.2534, (e) 26.4736/0.1549, (f) 26.6683/0.1397, and (g) 26.0073/0.1356.

SRs including SRGAN [11] and ESRGAN [21]. Table 1 lists the LPIPS, PSNR, and SSIM values for the three test datasets. As shown in the table, EDSR and SRResNet have the highest PSNR and SSIM scores in all datasets, whereas SRGAN, ESRGAN, and RBDN are not prominent in this type of distortion-based metric. When looking closely, we found that SRGAN, ESRGAN, and RBDN alternately achieved the highest PSNR and SSIM scores; thus, these two metrics still show similar scores.
better textures, such as the birds in Fig. 6 with a well-layered feather texture, for Fig. 7 where we restored the straw more clearly, and Fig. 9, where the plants recovered are the most in line with the real-world plant effect. Compared with other SR methods, the structure in our results is clear and there are no serious distortions, whereas the other methods fail to give the object a satisfactory appearance. A qualitative comparison verifies that our proposed RBDN method can learn more about structural information in a gradient space, which helps to generate realistic SRs by saving images of the geometry.

At the same time, our model exhibits a more impressive convergence speed. ESRGAN took 6 min and 23 s for the first 500 iterations during the pre-training process, whereas our model took only 5 min and 19 s in the same environment. While obtaining good visuals, faster convergence speeds are achieved, which means that when we train deeper network models, the training time is greatly reduced.

V. CONCLUSION

We found that a BN layer is used by most of the networks currently used to deal with SR reconstruction problems. Although it can speed up the training and convergence speed and prevent problems such as a gradient disappearance and overfitting, the problem of artifacts that the BN layer may bring to the final result cannot be ignored. At the same time, we also found that the use of dense cascading for subsequent layers avoids a direct summation to maintain the features of the previous layer. Thus, the problem of image sharpening can be improved. Inspired by these insights, we removed the BN layer and introduced a dense cascade. Simultaneously, considering the adverse effects of removing the BN layer and the convergence speed of the model, we replaced the activation function and added a bottleneck structure. In the course of the experiment, we found that the qualitative and quantitative indicators obtained by our model on the BSD100, Set5, and Set14 datasets achieve a better performance than the results obtained by commonly used networks. RBDN performs better than the existing commonly used models and minimizes the resource consumption caused by the training models. Future studies will explore the RBDN characteristics in the following ways: (1) research will be conducted a more lightweight design of an RBDN and (2) we will attempt to use an RBDN under a wider range of application scenarios, such as image classification and semantic segmentation.

ACKNOWLEDGMENT

(Zeyu An and Junyuan Zhang contributed equally to this work.) The statements made herein are the sole responsibility of the authors.

REFERENCES

[1] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2014, pp. 184–199.
[2] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, pp. 295–307, Feb. 2016.
[48] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, “Memory-efficient implementation of DenseNets,” 2017, arXiv:1707.06990. [Online]. Available: http://arxiv.org/abs/1707.06990

[49] P. Shamsolmoali, M. Zareapoor, E. Granger, H. Zhou, R. Wang, M. E. Celebi, and J. Yang, “Image synthesis with adversarial networks: A comprehensive survey and case studies,” Inf. Fusion, vol. 72, pp. 126–146, Aug. 2021.

ZEU AN is currently pursuing the bachelor’s degree with Southwest University, Chongqing, China. His major concern is information security. His research interests include cyberspace security, residual networks, super-resolution reconstruction, and visual SLAM.

JUNYUAN ZHANG is currently pursuing the bachelor’s degree with Southwest University, Chongqing, China. His major concern is information security. His research interests include cyberspace security, residual networks, super-resolution reconstruction, and machine olfaction.

ZIYU SHENG is currently pursuing the master’s degree with Southwest University. His research interest includes applications of neural networks in the field of load forecasting.

XUANHE ER is currently studying with the Chengdu University of Information Technology, Sichuan, China. His major is data science and big data technology. His current research interests include deep learning, data science, and image processing.

JUNJIE LV is currently pursuing the M.Sc. degree with Southwest University. His research interests include deep learning and its applications in computer vision.

* * *