Adaptive Residual Networks for High-Quality Image Restoration

Yongbing Zhang, Lulu Sun, Chenggang Yan, Xiangyang Ji, and Qionghai Dai, *Senior Member, IEEE*

***Abstract*— Image restoration methods based on convolutional neural networks have shown great success in the literature. However, since most of networks are not deep enough, there is still some room for the performance improvement. On the other hand, though some models are deep and introduce shortcuts for easy training, they ignore the importance of location and scaling of different inputs within the shortcuts. As a result, existing networks can only handle one specific image restoration application. To address such problems, we propose a novel adaptive residual network (ARN) for high-quality image restoration in this paper. Our ARN is a deep residual network, which is composed of convolutional layers, parametric rectified linear unit layers, and some adaptive shortcuts. We assign different scaling parameters to different inputs of the shortcuts, where the scaling is considered as part parameters of the ARN and trained adaptively according to different applications. Due to the special construction of ARN, it can solve many image restoration problems and have superior performance. We demonstrate its capabilities with three representative applications, including Gaussian image denoising, single image super resolution, and JPEG image deblocking. Experimental results prove that our model greatly outperforms numerous state-of-the-art restoration methods in terms of both peak signal-to-noise ratio and structure similarity index metrics, e.g., it achieves 0.2–0.3 dB gain in average compared with the second best method at a wide range of situations.**

***Index Terms*— Adaptive residual network, image denoising, image super resolution, JPEG image deblocking, high-quality.**

I. Introduction

IMAGE restoration, a classical and fundamental problem, aims at recovering the latent clean image y from its degraded observation x, which may be produced by noise contamination introduced during the image acquisition or transport over analog media. For the sake of simplicity, we commonly assume the corruption factor to be a linear operator $\boldsymbol{A}$ combined with an additive white Gaussian noise (AWG) $\boldsymbol{N}$ of standard noise deviation. In other words, the measurement could be formulated as $x = \boldsymbol{A}y + \boldsymbol{N}$. One intuitive way of image restoration is to obtain an estimate of y by minimizing the objective function as follows.

$$\hat{y} = \min_{y} \|x - Ay\|_2^2 . \tag{1}$$

The analytical solution of Eq.(1) can be expressed as $\hat{y} = (A^T A)^{-1} A^T x$. However, the solution exists only if A is invertible, which causes the image restoration to be an ill-posed problem.

To address this problem, traditional image restoration methods usually employ regularization techniques by adding some constraints to derive the estimation in Eq.(1). As a basic regularization theory, Tikhonov and Arsenin [1] first proposed an image restoration method by adding a regularization term as follows:

$$\hat{y} = \min_{y} \|x - Ay\|_2^2 + \gamma \, \|\Phi y\|_p^2 , \tag{2}$$

where γ is the regularization term balancing the tradeoff between the error and the smoothness of the data, Φ represents the penalty term, and p is the norm of the penalty function. Generally, p is usually set to 2 or 1, which represents the classical L2 or L1 norm techniques used for image restoration. Since L2 norm technique usually results in a smoothing effect on the restored image [2], especially at the edge regions, the majority traditional image restoration methods are based on L1 norm regularization. The simplest form of penalty term Φ in L1 norm regularization problem is $\Phi = I$, which is the well-known least absolute shrinkage and selection operator (LASSO) method [3]. In addition, the penalty term Φ can also be $\Phi = \nabla$, becoming the famous total variation (TV) [2], [4] method, which is able to preserve the edge information in the restored image. Furthermore, we can also add more constraints in the objective function in Eq.(2), such as the sparse representation methods [5]–[7] and low-rank minimization methods [8], [9]. Based on the assumption that each patch of an image can be accurately represented by a few elements from a basis, the sparse model has achieved a great success in image restoration [5]–[7]. Low-rank minimization is another strategy to represent similar patches as a matrix with each column being a stretched patch vector, and exploit the low-rank prior of the matrix for image restoration. Dong *et al.* [8] proposed a low-rank minimization approach

Manuscript received June 19, 2017; revised December 15, 2017; accepted February 18, 2018. Date of publication March 5, 2018; date of current version April 6, 2018. This work was supported in part by the Projects of International Cooperation and Exchanges NSFC under Grant 61620106005, in part by the National Science Foundations of China under Grant 61571254, in part by the Guangdong Special Support Plan under Grant 2015TQ01X16, and in part by the Shenzhen Fundamental Research Fund under Grant JCYJ20160513103916577. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Khan M. Iftekharuddin. *(Corresponding author: Chenggang Yan.)*

Y. Zhang is with the Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China (e-mail: zhang.yongbing@sz.tsinghua.edu.cn).

L. Sun is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: sunll15@mails.tsinghua.edu.cn).

C. Yan is with the Institute of Information and Control, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: cgyan@hdu.edu.cn).

X. Ji and Q. Dai are with TNLIST, Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: xyji@tsinghua.edu.cn; qhdai@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2812081

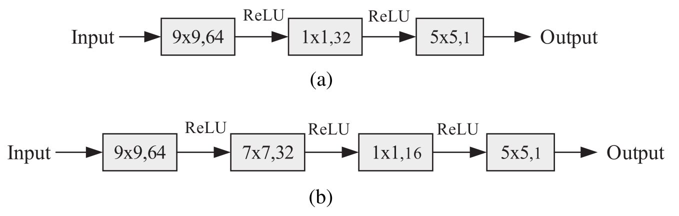


Fig. 1. Comparisons of SRCNN and ARCNN. (a) Architecture of SRCNN [10]. (b) Architecture of ARCNN [11].

based on nonlocal similarity and achieved highly competent performance. Besides, a weighted nuclear norm minimization (WNNM) [9] was also proposed by conducting weighted singular value decomposition to the stacked matrix. Since the singular values can be assigned different weights, WNNM preserves much better local image structures, leading to superior performance in terms of both objective and subjective criterions.

Recently, due to the immerse popularity of deep learning, a lot of models using neural networks have shown great potential in image restoration. The first successful attempt should be considered as denoising auto-encoder (DA) [12], which is actually a two-layer neural network that tries to reconstruct the original input from a noisy version of it. Let x be the corrupted input image vector with length d and y of the same length be the corresponding estimated clean version. DA could be formulated as:

$$y = \sigma(b_2 + W_2\sigma(b_1 + W_1 x)), \tag{3}$$

where $\sigma(x) = (1 + exp(-x))^{-1}$ is the sigmoid activation function applied element-wise to vectors, W_1 is a $d' \times d \in \mathbb{R}$ weight matrix with d' being the length of hidden representation, $W_2 = W_1^T$, and $b_i(i = 1, 2)$ represents the biases respectively. Instead of using such a two-layer architecture, Burger *et al.* [13] put forward a new network to build the connections between degraded images x and their corresponding clean ones y with plain multi-layer perception (MLP). The connections could be represented by a nonlinear function, which can be written as

$$y = b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)), \tag{4}$$

where $W_i(i = 1, 2, 3)$ is the weight matrix, $b_i(i = 1, 2, 3)$ is the vector-valued bias, and tanh function operates componentwise. Here W_1 is a $d' \times d \in \mathbb{R}$ matrix, W_2 is $d'' \times d' \in \mathbb{R}$ and W_3 is $d \times d'' \in \mathbb{R}$, with d' and d'' being the length of hidden representations, respectively.

With the wide application of convolutional neural networks [14]–[17], more effective image restoration methods have been put forward in recent years, among which the most famous one is super-resolution convolutional neural network (SRCNN) [10]. SRCNN is a three-layer (Namely, feature extraction, non-linear mapping, and reconstruction layer) convolutional neural network for image super-resolution, in which the filters of spatial size were 9×9, 1×1, and 5×5 respectively, as can be seen in Fig. 1(a). The network can be expressed as

$$y = b_3 + W_3 * f(b_2 + W_2 * f(b_1 + W_1 * x)), \tag{5}$$

where $W_i(i = 1, 2, 3)$ is the weight matrix, $b_i(i = 1, 2, 3)$ is the vector-valued bias, and f is the rectified linear unit (ReLU, $\max(0, x)$). Remarkably, "*" denotes the convolution operation. W_1 corresponds to n_1 filters of support $c \times f_1 \times f_1$, where c is the number of channels in the input image, f_1 is the spatial size of a filter. Similarly, W_2 contains n_2 filters of size $n_1 \times f_2 \times f_2$ and W_3 has n_3 filters of size $n_2 \times f_3 \times f_3$.

Based on SRCNN, artifacts reduction convolutional neural networks (ARCNN) [11] was also proposed for compression artifacts reduction. As can be seen in Fig. 1(b), ARCNN is actually a four-layer convolutional neural network, which employs an additional feature enhancement layer, compared to SRCNN, to improve the mapping accuracy.

Although these models based on neural networks have obtained excellent restoration results, there are still some shortcomings could be improved. For example, MLP is just a plain fully connected neural network, and it does not contain any convolutional layers. While SRCNN and ARCNN both contain convolutional layers, they are only tasked with specific applications and difficult to extend to others. What's more, these neural models are merely made up of one fully connected network, without considering introducing shortcuts, which is very useful and effective in high-level image processing, such as image recognition.

Shortcut, or called skip connection, is firstly presented by He *et al.* [18] for image recognition. In this case, shortcuts are simply identity mapping, adding to the later layers. The goal is to speed up the training process without increasing the number of parameter or computational complexity. What is more, because of such a useful skill, these models obtain better performance.

Motivated by the observations above, in this paper, we propose a high-quality image restoration method by devising an adaptive residual network (ARN) to build more precise connections between the corrupted images and their corresponding original ones. Firstly, we build a deep convolutional neural network to map a degraded image to its latent high-quality one via a supervised learning process. We utilize the mean square error (MSE) loss between the predicted clean image and the true original image to determine our ARN. The most important step is that we put adaptive and appropriate shortcuts to the deep networks. In our model, the short cuts can change their scaling flowing to the next layers, other than merely identity linear to connect two layers. Apparently, the introduction of scaling flowing is the biggest difference when comparing our shortcut to the shortcut in [18]. Our shortcut could pass more detailed information flowing across several layers without attenuation, fasten the convergence and reduce the loss when training our ARN. Furthermore, such a residual architecture is proved to be able to extract more useful features from the input images, which is very effective in restoring better details of high quality images. In detail, appropriate layers of 3×3 convolutional layers could extract enough features from the degraded images. We also add the last 1×1 convolutional layer, which plays an important role as well. It could strengthen the power of our network, bringing about better performance. Last but not least, xavier initialization [19], adaptive moment estimation (adam) [20] optimizer, parametric rectified linear unit (PReLU) [21], and other training strategies are employed to speed up the converging process and achieve global optimal values. Except the architecture of neural network, the dataset for training also plays an important role. In image restoration, there are some recognized and public datasets for different applications, such as BSD500 [22], 91images [7], set5 [23] and so on. These training images are all high-quality without any noise or blur, which is helpful to learn useful priors. For the sake of good performance and fair comparison, we select the publicly well known BSD500 [22], 91images [7] and set5 [23] as our training dataset in this paper.

Overall, the main contributions of our work can be summarized as follows:

1) We propose an adaptive residual network (ARN) for high-quality image restoration. Our ARN builds very accurate mappings between the degraded images and their corresponding clean ones. Solely due to the introduction of adaptive and appropriate shortcuts, could our ARN give great performance in image restoration.

2) Our ARN can handle many image restoration applications, such as Gaussian image denoising, single image super resolution, image deblocking, and so on at the same time. Compared to existing networks, which could only solve one or two problems, our ARN are applicable in different problems by just changing the corresponding training datasets.

3) Considering the difficulty of training deep networks, we adopt some effective strategies, such as PReLU [21], xavier initialization [19], adaptive moment estimation (adam) [20] optimizer, and other training strategies to speed up the training procedure and obtain global optimal values.

The remainder of this paper is organized as follows: In Section II, we describe several related works about image restoration. Section III introduces the details of our ARN and makes some comparisons on various residual units and network architectures. The analysis of scaling parameters within the proposed shortcuts in different applications is provided in Section IV. Experiments are brought in Section V, showing great improvements of the proposed idea in Gaussian image denoising, single image super resolution, and image deblocking. Section VI concludes the paper with discussions related to future work.

II. Related Work

Existing image restoration approaches are usually based on traditional algorithms and network-based models. The former methods include image priors [24]–[27], example-based [28]–[30], dictionary-based [6], [31], transformed domain [32]–[34], and statistical natural models [35]–[37] *et al.* However, the latter network-based methods draw more attention and produce state-of-the-art performance. We will analyze some representative methods for Gaussian image denoising, single image super resolution and JPEG image deblocking from the two aspects.

In Gaussian image denoising, the most famous method is block matching and 3D filtering (BM3D) [25], which combined nonlocal self-similarity with an enhanced sparse representation in transformed domain. In addition, based on the general prior knowledge that larger singular values of the patch matrices of original image are more important than the smaller ones, WNNM [9] achieved great success in image denoising. From the view of deep learning, one early try was DA, which could predict the uncorrupted data based on the corrupted data point. Stacking these DAs to form a deep network by feeding output of the previous layer to the current one as input was the main idea in stacked denoising auto-encoder (SDA) [12]. Furthermore, Burger *et al.* [13] attempted to learn a mapping from a noisy image to a noise-free one directly with MLP applied to image patches. Jain and Seung [38] carried out the first successful try of image denosing based on convolutional neural network, which offered similar performance in the blind denoising setting as compared to other techniques in the non-blind setting. However, no matter using either fully plain frameworks or convolutional architectures, the depth of these networks are too shallow to extract enough useful information for reconstructing high-quality images.

For image super-resolution, one famous benchmark is anchored neighborhood regression (ANR) [28], which learned sparse dictionaries and regressors anchored to the dictionary atoms and precomputed the corresponding embedding matrix. Similarly, simple functions (SF) [29] also provided state-of-the-art quality performance. But it relied on clusters and corresponding learned functions. To be specific, it split the feature space into numerous subspaces and collected sufficient training exemplars to learn simple regression functions. Ingeniously, A+ [30], combining the advantages of ANR and SF, built on the features and anchored regressors from ANR but learned the regressors using full training material, which is the same as SF. Hence, it obtained better quality and lower time complexity. Resembling to the neighbor embedding methods, jointly optimized regressors (JOR) [31] is another effective and computationally efficient algorithm. It jointly learned a fixed number of regressors, which could be selected adaptively for the appropriate input image. Besides the above mentioned classic conventional image super-resolution methods, the deep learning based methods are also shown great potentials recently. As mentioned in Section I, the most famous deep learning method is SRCNN [10]. It firstly utilized few convolutional and other layers to learn the mappings from low-resolution images to high-resolution ones. Though SRCNN already achieved some pleasant results, it still has a lot of room to improve the performance.

Referring to JPEG image deblocking, there are also numerous methods, focusing on removing deblocking and ringing artifacts caused by the lossy quantization of transform coefficients. There are larger numbers of algorithms based on filtering in the DCT-domain [33], [34], [39], [40]. Among these approaches, Zhang's method [34] reduced compression artifacts by optimizing overlapped block transform coefficients, which were estimated from non-local blocks of original images. Besides, network-based model also immersed recently. ARCNN [11] mentioned in Section I showed great performance by extracting features from large dataset. In spite of some differences between ARCNN and SRCNN, as shown in Fig. 1(a) and Fig. 1(b), the weaknesses of them are a little alike.

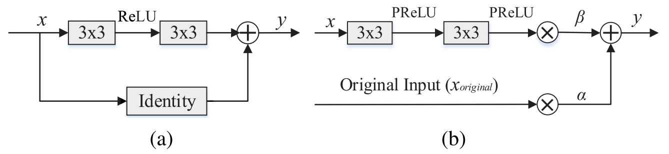


Fig. 2. Comparisons of residual units. (a) Residual unit in [18]. (b) Adaptive residual unit (ARU).

In our work, we build adaptive residual network (ARN) with changeable skip connections for image restoration. Such a trainable residual architecture is useful to reconstruct more vivid images. Namely, adaptive shortcuts are very helpful to pass specific details from the original image to the end convolutional layers without adding computational cost. Moreover, we investigate the property of different residual units and depth of neural networks, both of which are very important to improve the performance of our model. In the end, we use some common strategies to train our ARN to make it converge faster and perform better.

III. Proposed Adaptive Residual Network (ARN)

In this section, we will first illustrate the details of our residual units including the interior structures and mathematical expressions. Next, we will demonstrate the structure of our network, containing the numerous layers of the network and the channels of each layer. The following will be the loss function employed. Finally, some useful techniques applied in the training of ARN model are provided.

A. Adaptive Residual Units (ARU)

Generally, the deeper the network is, the more power it will have [18], [41]. However it will be more difficult to train such a deep network because of vanishing gradients [42]. Hence, to address the degradation problem, He *et al.* [18] put forward the residual learning framework.

As seen in Fig. 2(a), the residual building block is made of two 3×3 convolutional layers, the corresponding ReLU, and one identity skip connection. Here, x and y are the input and output vectors of the layers considered. Formally, the building block can de defined as:

$$y = F(x, W_i) + x. \tag{6}$$

Here $F = W_2\sigma(W_1x)$ in which σ denotes ReLU [43]. Obviously, such shortcuts connections introduce neither extra parameter nor computation complexity.

Similarly, our residual unit, as shown in Fig. 2(b), can be formulized as:

$$y = \beta F(x, W_i) + \alpha x_{original}, \tag{7}$$

where $x_{original}$ indicates the original input vector, α and β represent the scaling factors of different inputs. The initial values of α and β are set to 0.5 and 0.25 based on experience. Then the two variables would be optimized together with the convolutional weights, biases, and other variables by adam [20] method to make the network converge to an optimal value.

It should be noted that the x in Fig. 2(a) and Fig. 2(b) both represents the input of current layer, in other words, it is the output of the adjacently previous layer. Therefore, such input x will change when the network gets deeper. However, $x_{original}$ will always be the input degraded image without altering. From Fig. 3, we could see there are six identity skip connections between the input low-quality image and the final output. And all the six $x_{original}$ are the same input while x is the output of previous ARU.

Comparing to the existing residual unit, our ARN block has three differences. Firstly, rather than using ReLU activation function, we use PReLU [21] for training, as can be seen in Fig. 4. The distinction between ReLU and PReLU will be demonstrated in the following sub-sections. Secondly, our shortcut connects the original input with the following stacked layers, rather than linking the outputs of the previous layers to the current one. This matters a lot for reserving the useful information of input. Thirdly, we introduce α and β to smartly balance the importance between the original input and the output of previous layer. In this paper, the scaling values of α and β are considered as part of the weighting parameters of the ARN model, and the optimal value can be obtained during the training process. To be noticed, we also tried setting the scaling parameter to be a constant value. But from the experimental results and theoretical analysis, we find that making the scaling parameter adaptive by the cases is very beneficial.

B. Network Architecture

As can be seen in Fig.3, our ARN is made up of one 3×3 convolutional layer, six ARUs, and one 3×3 convolutional layer sequentially. We usually put a PReLU layer after the convolutional layer, which have been omitted in the figure for simplicity, and the reason why we use PReLU rather than ReLU will be illustrated in the training strategies section. The first 3×3 convolutional layer is used to extract the features from the input low-quality images. Then, the extracted features will be sent to the six ARUs. After that, the last 3×3 convolutional layer works as reconstructing high-quality images. The numbers after 3×3 in Fig.3 represent the output channels of current layers. Each ARU has two 3×3 convolutional layers, and for example, (64,128) in the 3rd ARU represents there are 64 channels in the first 3×3 convolutional layer and 128 channels in the second 3×3 convolutional layer. Facts proved that 3×3 convolutional layers [44] have sufficient ability to extract good features as long as the network is deep enough. Furthermore, they will produce fewer parameters than larger convolutional kernels, which will reduce computation complexity greatly.

Especially, the part that our ARN differs heavily from conventional residual networks is the six ARUs. From Fig. 2(b), we know that all the six ARU blocks introduce the original input to the current layers, and the scaling parameters α and β are decided by the network itself and specified applications. What is the relationship between scaling and application can be found in Section IV.

C. Loss Function

We adopt MSE, the most common method to measure the differences between two images, as the loss function in

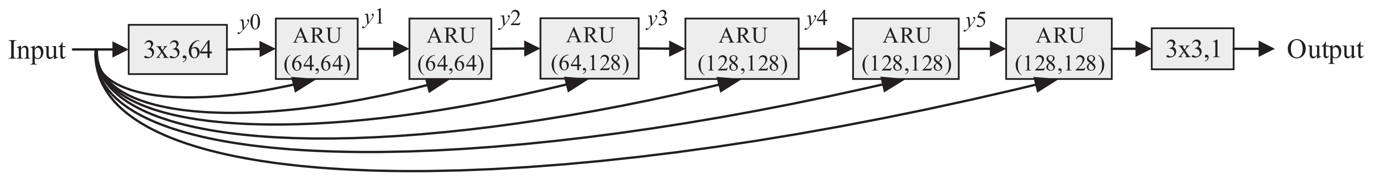


Fig. 3. The architecture of our ARN. 3×3 denotes the filter size of convolutional layers. The numbers after 3×3 represent the output channels of current layers. One ARU has two 3×3 convolutional layers, and for example, $(64, 128)$ in the 3rd ARU represents there are 64 channels in the first layer and 128 channels in the second layer. Each PReLU after the convolutional layer has been omitted for limited space. In addition, the six identity shortcuts between input and each ARU corresponds to the original input channels in Fig. 2(b).

our model. Learning the end-to-end function F from low-quality images to its high-quality counterparts needs to estimate the weights θ represented by the convolutional kernels as well as the scaling parameters of the shotcuts. This is achieved by minimizing the MSE between the outputs of the network and the original image. More specifically, given a collection of n training image pairs x_i, y_i, where x_i is a low-quality image and y_i is the high-quality version as the ground truth, we minimize the objective function:

$$L(\theta) = MSE = \frac{1}{n}\sum_{i=1}^{n} \| F(x_i, \theta) - y_i \|_2^2 . \tag{8}$$

We record peak signal-to-noise ratio (PSNR)(dB) values on the testing set during training for different applications from Fig. 5(a)-(c). It can be observed that at the beginning, all the curve jitter heavily. But at last they converge to a straight line slowly.

D. Training Strategies

Lots of successful approaches have been proposed for training deep neural networks in the past few years [19]–[21], [45], [46]. Based on those achievements, in our work, as for activation, initialization, and optimization, we apply PReLU, xavier initialization, and adam optimization separately.

1) Parametric Rectified Linear Unit (PReLU): ReLU has been a common activation function used in deep learning after traditional sigmoid-like units. For ReLU, if the input is less than zero, the result is zero, which is very helpful to generate a sparse representation. For neural networks, the values of the activation layers can be viewed as a sparse representation of the input. Hence, ReLU can accelerate the process of network convergence and achieve better solutions. This is why an increasing number of researchers prefer to using ReLU other than conventional sigmoids.

From the comparisons in Fig. 4, in the negative part, PReLU introduces one learnable parameter. Obviously, it is a general form of ReLU. As stated in [21], the slope $\boldsymbol{a}$ in PReLU is an adaptively learned parameter that can offset the positive mean of ReLU, making it a little symmetric. In addition, experiments also prove that PReLU converges faster than ReLU and obtains better performance. So, we apply PReLU in our ARN.

2) Xavier Initialization: When creating neural networks, it is important for us to make choices for the initial weights and biases, especially on the occasion where the networks are very deep. In our model, we use the xavier [19] way of setting our initial weights and biases to help our neural networks learn faster. Xavier is to initialize the weights by drawing them from a distribution with zero mean and a specific variance. Glorot and Bengio [19] recommended using the following variance:

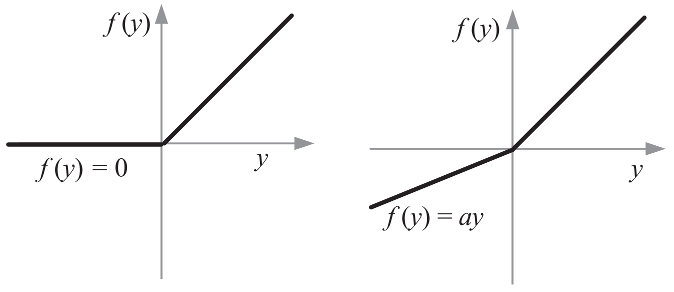


Fig. 4. ReLU vs. PReLU.

$$Var(W) = \frac{2}{n_{in} + n_{out}}, \tag{9}$$

where W is the initialization distribution for the neuron, n_{in} is the number of neurons feeding into it, and n_{out} is the number of neurons the result is fed to. The distribution used is typically Gaussian or uniform.

Experimental results verify that xavier strategy is better than the common prescription that chooses both the weights and biases using independent Gaussian random variables, normalized to have mean 0 and standard deviation 1.

3) Adam Optimization: Neural networks are often trained stochastically, in other words, using a method where the objective function changes at each iteration. By investigating some most commonly used gradient descent optimization algorithms [47]: stochastic gradient descent (SGD), momentum, nesterov accelerated gradient, adagrad, adadelta, rmsprop, and adam [20], we find that adam technique performs best in image restoration among all the algorithms, which is closely related to its computational principle. It computes a decayed moving average of the gradient and squared gradient (first and second moment estimates) at each time step. First, the first order moment moving average coefficient is decayed over time. Second, because the first and second order moment estimates are initialized to zero, some bias-correction is used to counteract the resulting bias towards zero. Thanks to these desirable properties, adam works well in practice and compares favorably to other adaptive learning algorithms.

IV. Analysis of Scaling Parameters Within Shortcuts

Based on the adaptive shortcuts architecture, we trained different models for the three representative image restoration

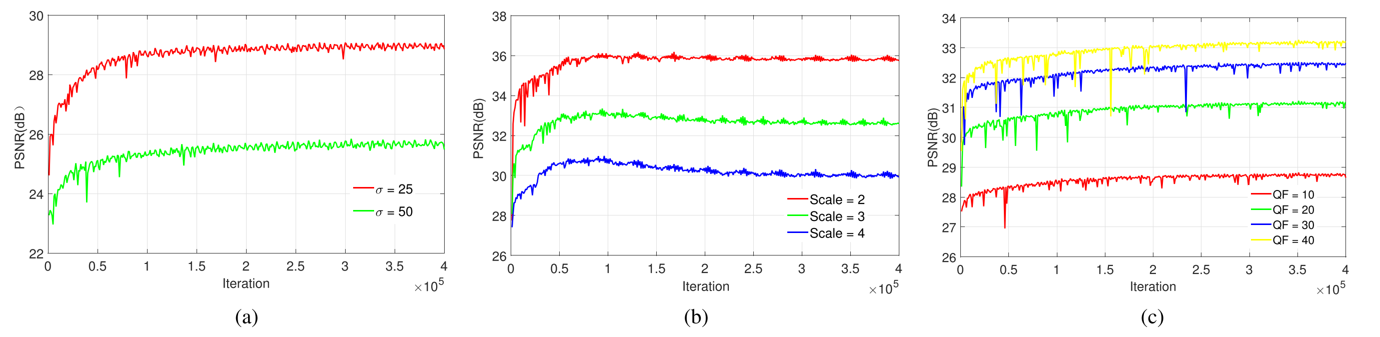


Fig. 5. PSNR(dB) values over the testing set during training for various applications. (a) Gaussian image denoising. (b) Single image super-resolution. (c) JPEG image deblocking.

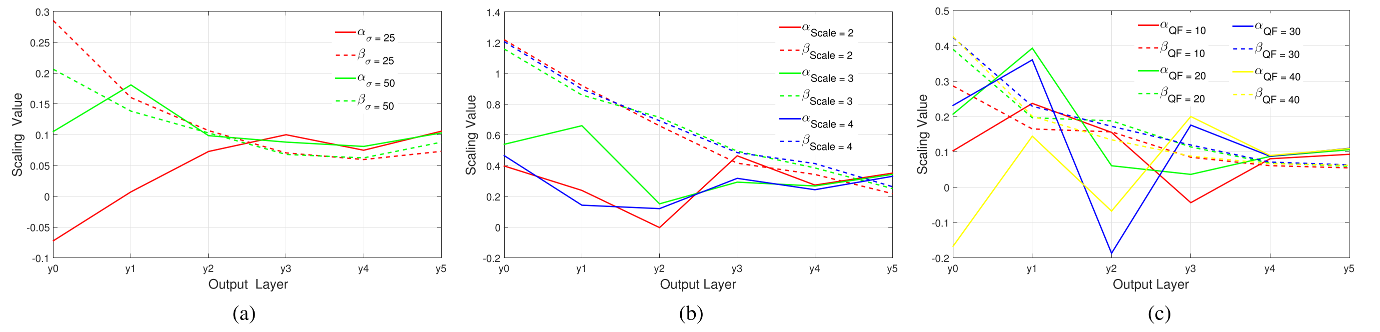


Fig. 6. Adaptive α and β of different layers for various applications. (a) Gaussian image denoising. (b) Single image super-resolution. (c) JPEG image deblocking.

applications: Gaussian image denoising, single image super-resolution, and image deblocking via changing the corresponding training and testing sets.

By observing the learned parameters α and β of differnt layers, we could get the following conclusions.

As can be seen in Fig. 6 (a), for Gaussian image deoising, in the superficial layers, α is very small while β is very large. As the network becomes deeper, α increases very slowly. At the same time, β decreases gradually. When noise level σ gets larger, α and β both start at a medium value. However, both α and β will reach unanimity at last layers. From the variation trend, we know that in shallow layers, the convolutional outputs play a more important role than the original input, because of Gaussian noises. And the gap will narrow when noise levels become larger. But at last, their importance seems to be the same.

From Fig. 6 (b), we can see the adaptive situation for single image super-resolution. Similarly, all αs start at a smaller value while βs are larger. Also, αs fluctuate greatly but βs change slowly. Likewise, αs and βs converge to a close value at later layers. All of these indicate that in the first layers, the extracted features are more important than the original images. Nonetheless, in the last layers, they almost become the same significance because of the more and more features.

The adaptive change trend for image deblocking is quite different from the case in image denoising and super-resolution, as shown in Fig. 6 (c). On one hand, βs still decrease very slowly. On the other hand, αs have a high fluctuation which is hard to find the rules. What we only could get is that all αs and βs converge to the same value at last, which also could be found in the above two applications.

In conclusion, all the three applications emphasize the importance of adaptive scaling between different layers and the original input. The applications decide the starting and ending values of αs and βs and the changing trends of them. But there also exists a constant law that all αs and βs go to a steady value around 0.1 at last. It seems to tell that different image restorations still have something in common, which is also the base of most methods.

V. Experiments

In this section, we perform experiments on three representative image restoration tasks: Gaussian image denoising, single image super-resolution, and image deblocking. By comparing our ARN with several state-of-the-art image restoration algorithms, we could see the huge advantage of our adaptive residual model in terms of PSNR and structural similarity index (SSIM) [48]. We only focus on the restoration of luminance channel (in YCrCb space) in this paper. Given a reference image f and a test image g, the SSIM can be defined as the following:

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g), \tag{10}$$

where

$$\begin{cases} l(f,g) = \dfrac{2\mu_f\mu_g + c_1}{\mu_f^2 + \mu_g^2 + c_1} \\ c(f,g) = \dfrac{2\sigma_f\sigma_g + c_2}{\sigma_f^2 + \sigma_g^2 + c_2} \\ s(f,g) = \dfrac{\sigma_{fg} + c_3}{\sigma_f\sigma_g c_3}. \end{cases} \tag{11}$$

Fig. 7. The testing set (gray-scale, 256×256), 15 images of which are widely used in image denoising.

TABLE I
PSNR (dB) RESULTS WITH DIFFERENT σ ON TESTING SET (SEE FIG. 7). THE BEST RESULT FOR EACH IMAGE IS **HIGHLIGHTED**

	$\sigma = 25$															
Method \ Image	Baboon	Boat	C.man	Couple	Hill	J.Bean	Lena	Man	Monarch	R.R.Hood	Paint	Parrots	Pentagon	Peppers	Starfish	Ave.
BM3D[21]	26.51	28.42	29.08	28.44	29.06	32.89	30.36	28.43	28.88	30.73	29.03	31.55	28.21	29.96	28.71	29.35
EPLL[31]	26.68	28.72	29.25	28.42	29.03	32.98	30.29	28.73	29.34	30.63	29.28	31.59	28.07	30.13	28.97	29.47
WNNM[9]	26.81	28.77	29.64	28.64	29.29	33.97	30.87	28.73	29.84	31.22	29.51	32.12	**28.72**	30.42	29.54	29.87
MLP[13]	26.87	28.85	29.30	28.75	29.31	33.64	30.79	28.80	29.62	30.93	29.65	32.10	28.23	30.17	29.26	29.75
PCLR[23]	26.75	28.88	29.68	28.73	29.30	34.03	30.78	28.74	29.75	30.98	29.58	32.01	28.42	**30.49**	29.35	29.83
ARN	**26.95**	**29.18**	**29.74**	**29.02**	**29.48**	**34.31**	**31.14**	**29.03**	**30.13**	**31.34**	**30.07**	**32.50**	28.61	30.36	**29.85**	**30.11**
	$\sigma = 50$															
Method \ Image	Baboon	Boat	C.man	Couple	Hill	J.Bean	Lena	Man	Monarch	R.R.Hood	Paint	Parrots	Pentagon	Peppers	Starfish	Ave.
BM3D[21]	24.07	25.25	25.98	25.17	26.22	29.43	27.05	25.55	25.62	27.75	25.66	28.32	25.35	26.60	25.44	26.23
EPLL[31]	24.13	25.48	26.04	25.09	26.14	29.18	26.93	25.59	25.76	27.58	25.88	28.00	25.02	26.64	25.50	26.20
WNNM[9]	24.21	25.54	26.45	25.27	26.35	30.11	27.47	25.71	26.32	28.13	25.97	28.71	**25.72**	26.95	26.03	26.60
MLP[13]	24.31	25.68	26.37	25.49	26.47	29.92	27.54	25.87	26.24	28.15	26.21	29.01	25.48	26.69	25.92	26.62
PCLR[23]	24.12	25.57	**26.60**	25.27	26.26	30.08	27.38	25.74	26.25	28.02	26.05	28.70	25.34	**27.07**	25.87	26.55
ARN	**24.35**	**25.91**	**26.60**	**25.54**	**26.51**	**30.15**	**27.63**	**25.96**	**26.73**	**28.37**	**26.47**	**29.12**	25.53	26.92	**26.27**	**26.80**

In Eq. (11), μ_f, σ_f, and σ_f^2 represent the average, standard deviation, and variance of image f, respectively, and the same as μ_g, σ_g, and σ_g^2. In addition, σ_{fg} means the covariance of f and g, and $c_i (i = 1, 2, 3)$ is the positive constant used to avoid a null denominator.

The implementations are all from the publicly available codes provided by the authors.[1] We use the deep learning library Tensorflow on an NVIDIA GTX TITAN X GPU with 3072 CUDA cores and 12GB of RAM to implement all the operations in our network.

A. Gaussian Image Denoising

The goal of image denoising is to recover the latent clean image from its corrupted version. For better dealing with such an ill-posed problem, we usually assume the corruption kernel as additive white Gaussian noise $\boldsymbol{N}$. So the noise-corrupted version x could be formulated as $x = y + \boldsymbol{N}$. We compare our ARN with several state-of-the-art denoising methods, including BM3D [25], EPLL [35], WNNM [9], MLP [13], and PCLR [27].

1) Training Details: We use Berkeley Segmentation Dataset BSD500 [22] as the training set and 15 widely used test images as the testing set (It can be found in Fig. 7). For increasing the training set, we segment these images to overlapping patches of size 50×50 with stride of 10.

2) Quantitative Results: We compute PSNR and SSIM to evaluate quantitatively the denoising results, which can be seen in Table I and Table II. From the comparing results on noise level $\sigma = 25$ and 50, we will have the following observations. Firstly, ARN holds the overwhelming superiority on PSNR than other state-of-the-art methods with the provided two noise levels in average. Especially, the superiority over the second best method reaches to 0.24 dB and 0.18 dB on $\sigma = 25$ and 50 respectively. It should be noticed that our method is always the best, but the second best changes when noise level varies. This reflects the robustness of ARN. Secondly, with regards to each image, ARN still has absolute advantage on PSNR. Among 15 images, ARN has the highest PSNR values on almost all images regardless of the increase of noise variance. Hence, no matter on the whole or individuals, our ARN both shows the great preponderance than other compared methods in terms of PSNR.

From Table II, it can be seen that ARN achieves significantly better SSIM indices than any other method. To be specific, ARN exceeds the second best method 0.0084 and 0.0067 on noise level $\sigma = 25$ and 50 respectively. Similar to the PSNR metric, under the metric of SSIM, our ARN always shows the best performance while the other methods shake when the noise level changes, which also claims the robustness of ARN.

3) Visual Quality: In addition to the objective measurements mentioned above, human subjective perceptivity is the ultimate judge of the image quality, which is also crucial to evaluate a denoising method.

To better show how much the high frequency information can be recovered, we use the very common canny method in MATLAB to perform edge detection and the results are depicted in Fig. 8. We could easily find that our method could retain the most precise high frequency information compared to the original clean image.

For further demonstrating the effectiveness of ARN, we show the visual comparisons from Fig. 9 to Fig. 10. On the whole, we can see that BM3D, EPLL, WNNM, MLP, and PCLR are all prone to over-smooth the images and lose some details, while ARN performs the best. Because of the accurate

[1]The source code of our ARN will be available after this paper is published.

TABLE II

SSIM (dB) RESULTS WITH DIFFERENT σ ON TESTING SET (SEE FIG. 7). THE BEST RESULT FOR EACH IMAGE IS **HIGHLIGHTED**

	$\sigma = 25$															
Image / Method	Baboon	Boat	C.man	Couple	Hill	J.Bean	Lena	Man	Monarch	R.R.Hood	Paint	Parrots	Pentagon	Peppers	Starfish	Ave.
BM3D[21]	0.7016	0.8084	0.8473	0.8218	0.7940	0.9350	0.8721	0.7972	0.8919	0.8080	0.8423	0.8837	0.7603	0.8605	0.8346	0.8306
EPLL[31]	0.7259	0.8228	0.8547	0.8166	0.7958	0.9336	0.8681	0.8062	0.8989	0.8091	0.8524	0.8790	0.7674	0.8664	0.8443	0.8361
WNNM[9]	0.7214	0.8164	0.8593	0.8263	0.7992	0.9513	0.8886	0.8070	0.9085	0.8198	0.8551	0.8928	0.7787	0.8728	0.8542	0.8434
MLP[13]	0.7260	0.8174	0.8555	0.8259	0.7994	0.9445	0.8809	0.8087	0.9003	0.8128	0.8574	0.8876	0.7652	0.8689	0.8485	0.8399
PCLR[23]	0.7176	0.8196	0.8619	0.8252	0.7981	0.9521	0.8850	0.8074	0.9082	0.8161	0.8571	0.8904	0.7700	0.8744	0.8517	0.8423
ARN	**0.7360**	**0.8304**	**0.8651**	**0.8372**	**0.8081**	**0.9580**	**0.8924**	**0.8182**	**0.9161**	**0.8244**	**0.8667**	**0.8996**	**0.7834**	**0.8788**	**0.8622**	**0.8518**
	$\sigma = 50$															
Image / Method	Baboon	Boat	C.man	Couple	Hill	J.Bean	Lena	Man	Monarch	R.R.Hood	Paint	Parrots	Pentagon	Peppers	Starfish	Ave.
BM3D[21]	0.5230	0.6864	0.7736	0.6810	0.6766	0.8839	0.7902	0.6880	0.8122	0.7377	0.7389	0.8241	0.5989	0.7860	0.7254	0.7284
EPLL[31]	0.5303	0.7019	0.7595	0.6720	0.6711	0.8601	0.7762	0.6865	0.8177	0.7277	0.7457	0.7991	0.5817	0.7856	0.7318	0.7231
WNNM[9]	0.5314	0.7005	0.7846	0.6852	0.6826	0.9054	0.8118	0.6973	0.8346	0.7501	0.7534	0.8377	0.6167	0.8005	0.7535	0.7430
MLP[13]	0.5406	0.7064	0.7933	0.6962	0.6872	0.9007	0.8055	0.7014	0.8283	0.7527	0.7614	0.8433	0.6090	0.7986	0.7501	0.7450
PCLR[23]	0.5279	0.7005	**0.7945**	0.6813	0.6732	**0.9112**	0.8093	0.6952	0.8361	0.7488	0.7572	0.8392	0.5952	**0.8081**	0.7471	0.7417
ARN	**0.5486**	**0.7144**	0.7933	**0.6974**	**0.6917**	0.9109	**0.8147**	**0.7067**	**0.8445**	**0.7548**	**0.7710**	**0.8466**	**0.6104**	0.8056	**0.7649**	**0.7517**

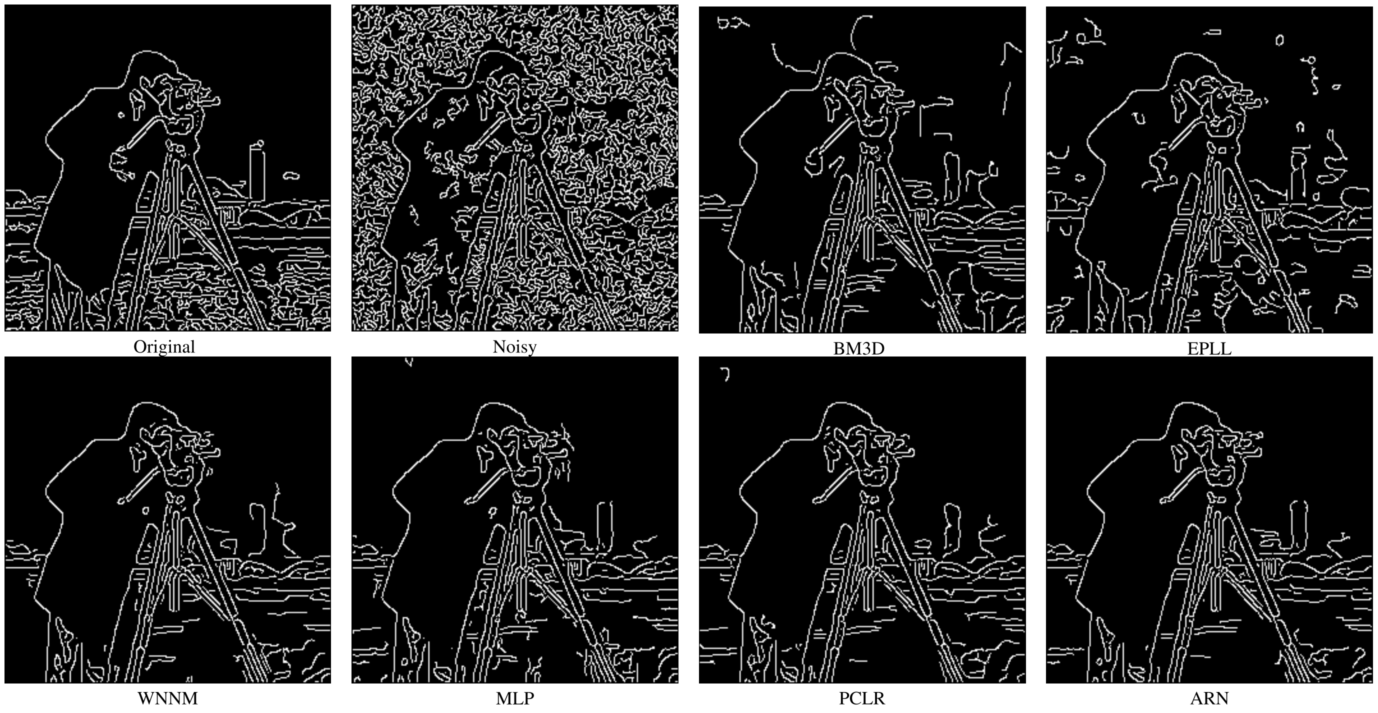


Fig. 8. Edge detection results on *Cameraman* with state-of-the-art methods ($\sigma = 50$).

mappings between the noisy images and the corresponding clean ones, our ARN seems to keep a good balance between noise removal and details preservation.

Particularly, from Fig. 9, it can be easily seen that BM3D loses the most detailed information on *boat*, especially on the red highlighted window, which is hard to see the fine lines behind the three thick steer structure lines. Though EPLL, WNNM, MLP, and PCLR could see a little such fine lines, they are still far from satisfaction. But interestingly, our ARN could reconstruct much better clear shape of these thick and fine lines.

As shown in the red highlighted regions of Fig. 10, one can clearly see that our ARN could recover the bridge of the nose and fluctuation lips on *Lena* simultaneously while the others miss a lot of details and textures. Focusing on the recovered lips, EPLL, MLP, and PCLR bring in a lot of impurities, consequently it is hard for us to recognize whether they are the real lips. Meanwhile, BM3D and WNNM lose much detailed information. In one word, they not only add too many artificial noises, but also over-smooth the edges and structures. On the contrary, our ARN reconstructs almost all the pleasant details.

Actually, we can get the same observations from the other images. It is obviously that with the increase of noise variance, other state-of-the-art methods either over smooth images, resulting in the loss of detailed information, or add too many artificial noises, which greatly deteriorates the image quality.

To conclude, thanks to the introduction of adaptive shortcuts into our network, ARN can learn the precise mappings from noisy images to the clean ones, producing much more visually pleasant denoising images.

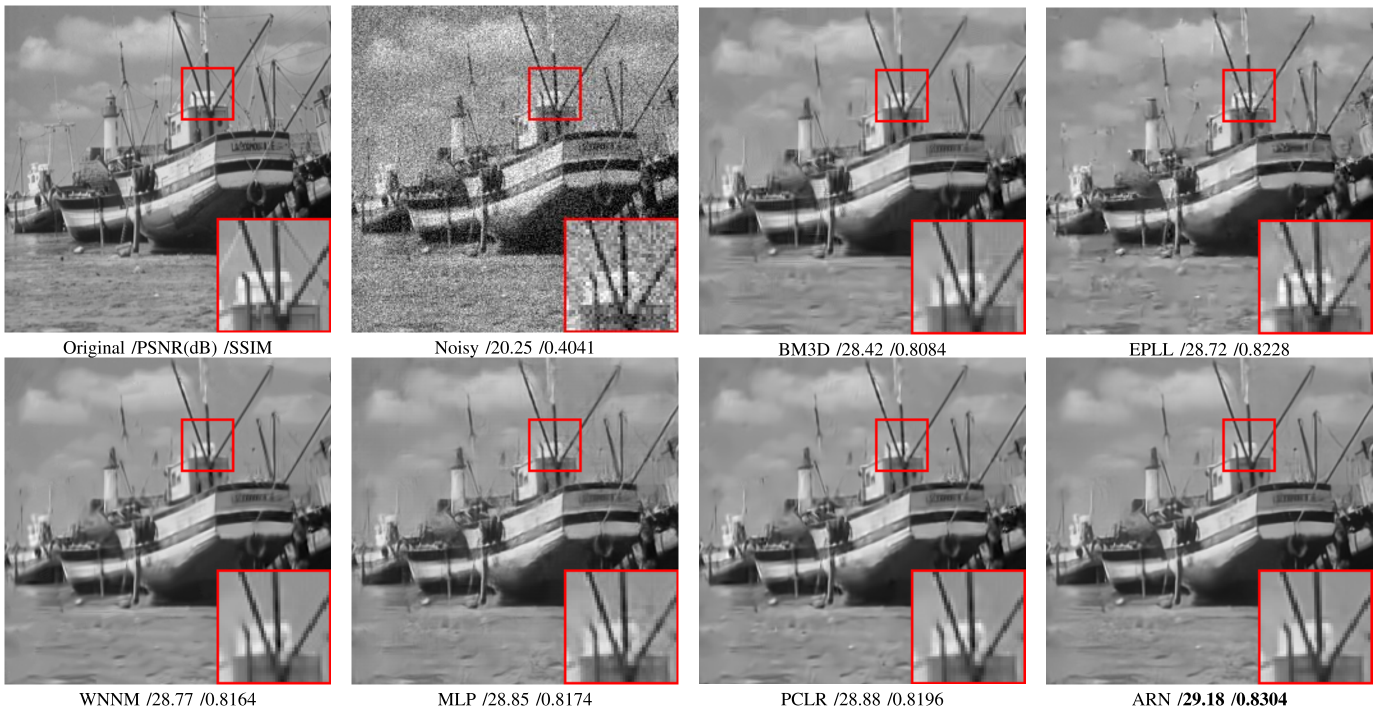


Fig. 9. Denoising comparison on *Boat* with state-of-the-art methods ($\sigma = 25$). (Zoom in for better view.)

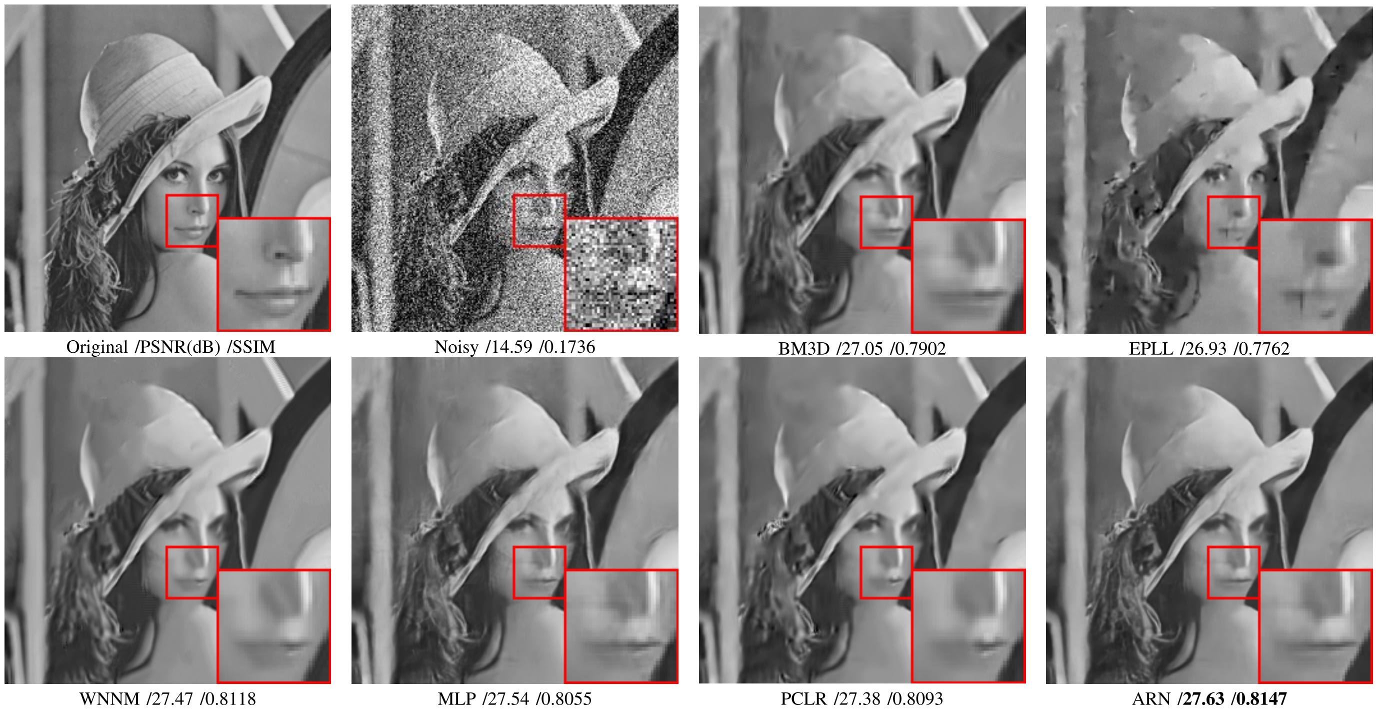


Fig. 10. Denoising comparison on *Lena* with state-of-the-art methods ($\sigma = 50$). (Zoom in for better view.)

4) Running Time: For a fair comparison, we profile time consumption of all the methods in a Matlab 2015b environment on the same PC (Intel CPU 3.30 GHz and 16GB RAM). Our method is implemented using GPU. From TABLE III, we could see that the running time spent on one image is the same by our method. However, the running time of all the other methods increases when σ gets larger, which revealed the absolute advantage of our ARN over other methods. This is mainly benefited from the architecture of residual learning and other training skills in the proposed method.

B. Single Image Super-Resolution

If we use $x = \boldsymbol{A}y$ to describe the relationship between high-resolution image y and low-resolution image x, and $\boldsymbol{A}$ is

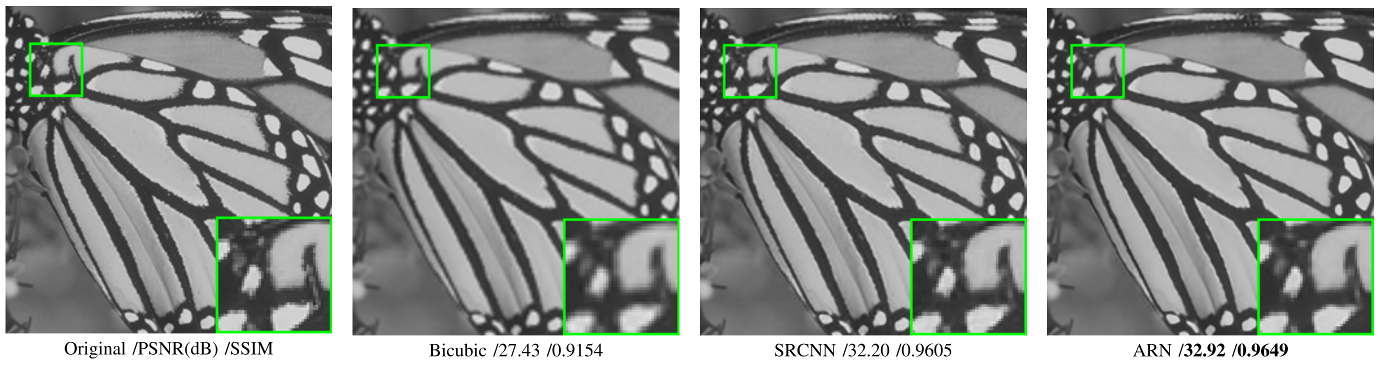


Fig. 11. Super-resolution comparison on *Butterfly* of set5 with state-of-the-art methods (Scale = 2). (Zoom in for better view.)

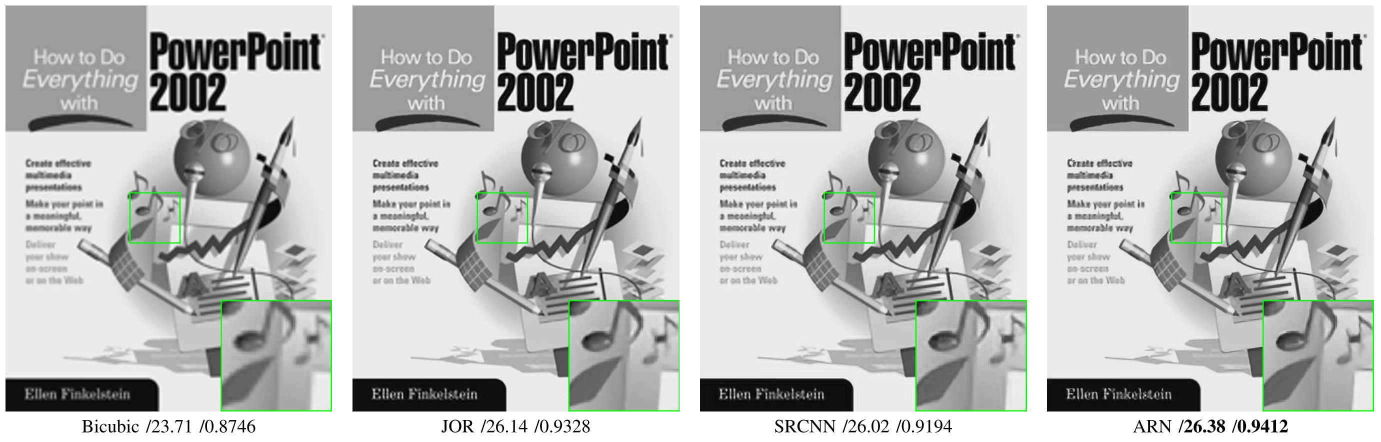


Fig. 12. Super-resolution comparison on *ppt3* of set14 with state-of-the-art methods (Scale = 3). (Zoom in for better view.)

TABLE III

AVERAGE RUNNING TIME(s) FOR ONE IMAGE WITH DIFFERENT NOISE LEVEL σ ON TESTING SET(SEE FIG. 7). THE BEST RESULT FOR EACH DATASET IS **HIGHLIGHTED**

σ	BM3D[21]	EPLL[31]	WNNM[9]	MLP[13]	PCLR[23]	ARN
$\sigma = 25$	**0.25**	31.20	176.33	3.63	63.52	0.80
$\sigma = 50$	1.41	32.43	242.25	24.57	167.55	**0.80**

TABLE IV

AVERAGE RESULTS OF PSNR(dB) ON THE SET5, SET14, BSD100, URBAN100 DATASET. THE BEST RESULT FOR EACH DATASET IS **HIGHLIGHTED**

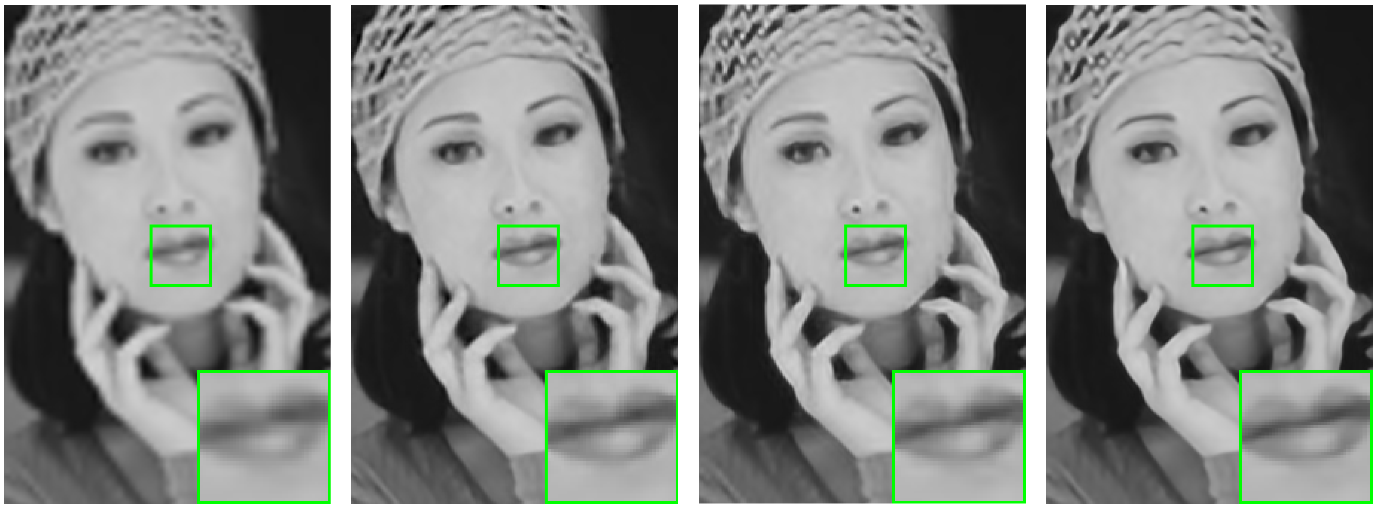
Method / Dataset	Bicubic	SRCNN[10]	JOR[27]	ARN
		Scale = 2		
Set5	33.66	36.34	-	**36.36**
Set14	30.23	**32.18**	-	32.17
BSD100	29.10	30.24	-	**30.44**
Urban100	26.52	28.20	-	**28.60**
		Scale = 3		
Set5	30.39	32.39	32.58	**32.66**
Set14	27.54	29.00	29.09	**29.09**
BSD100	27.01	27.92	27.98	**28.02**
Urban100	24.30	25.57	25.68	**25.85**
		Scale = 4		
Set5	28.42	30.08	30.19	**30.37**
Set14	26.00	27.20	27.26	**27.40**
BSD100	25.82	26.53	26.61	**26.69**
Urban100	23.03	23.98	24.11	**24.24**

a down-sampling factor, problem of recovering y from x could be viewed as image super-resolution. By comparing our ARN with classic network-based SRCNN [10] and JOR [31], we can see the great performance of ARN.

1) Training Details: For a fair comparison, following [10] and [31], we use 91 images from Yang *et al.* [7] as the training set and evaluate all the methods on the standard Set5 [23], Set14 [49], BSD100 [22], and Urban100 [50] datasets. Similar to [10], we first down-sample each image in the training set via scaling factors of $\times 2$, $\times 3$, and $\times 4$, without adding noise in the down-sampled images. Then, we train our models respectively for different scaling factors of $\times 2$, $\times 3$, and $\times 4$. Besides, we split training images into overlapping patches of size 33×33 with stride of 10 for increasing the datasets.

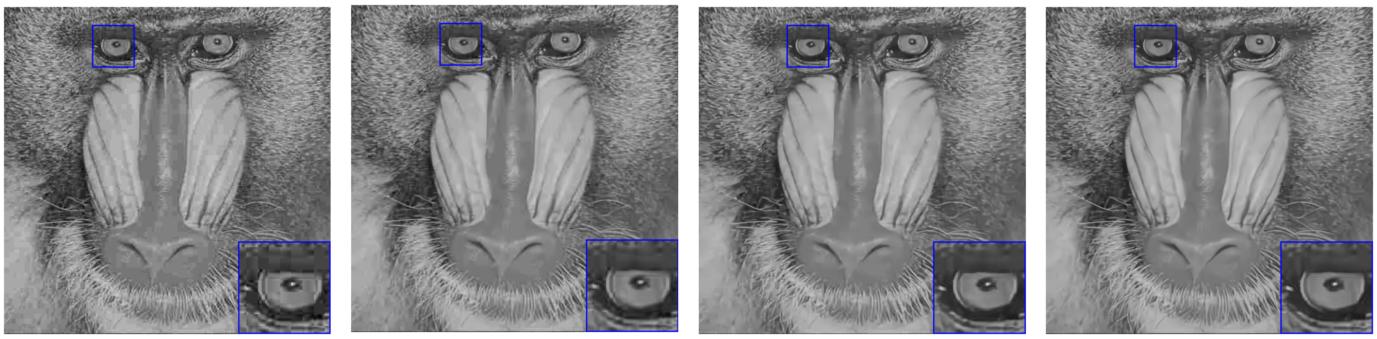
2) Quantitative Results: The comparing results, when scale = 2, 3, and 4, are shown in Table IV and Table V. Because JOR [31] does not contain trained model when scale = 2 in its code, we neglect the result for that case. It can be observed that our ARN always exceeds other competing methods a lot for all the down-sampling scales over various testing results.

3) Visual Quality: Apart from PSNR and SSIM, we also make visual quality comparisons to demonstrate the strong power of our ARN. We randomly select three common images



Bicubic /26.46 /0.8318 JOR /28.53 /0.8800 SRCNN /28.20 /0.8709 ARN **/28.65 /0.8830**

Fig. 13. Super-resolution comparison on *Woman* with state-of-the-art methods (Scale = 4). (Zoom in for better view.)



JPEG /24.33 /0.6732 Zhang's method /24.80 /0.6738 ARCNN /24.82 /0.6756 ARN **/24.97 /0.6863**

Fig. 14. Deblocking comparison on *Baboon* with state-of-the-art methods (QF = 10). (Zoom in for better view.)

TABLE V

Average Results of SSIM on the Set5, Set14, BSD100, Urban100 Dataset. The Best Result for Each Dataset Is **Highlighted**

Dataset \ Method	Bicubic	SRCNN[10]	JOR[27]	ARN
		Scale = 2		
Set5	0.9299	**0.9521**	-	0.9483
Set14	0.8687	**0.9039**	-	0.8988
BSD100	0.8267	0.8610	-	**0.8624**
Urban100	0.8242	0.8655	-	**0.8750**
		Scale = 3		
Set5	0.8682	0.9033	0.9079	**0.9085**
Set14	0.7736	0.8145	**0.8184**	0.8166
BSD100	0.7308	0.7700	**0.7732**	**0.7732**
Urban100	0.7258	0.7756	0.7836	**0.7898**
		Scale = 4		
Set5	0.8104	0.8530	0.8580	**0.8633**
Set14	0.7019	0.7413	0.7474	**0.7492**
BSD100	0.6625	0.6957	0.7008	**0.7036**
Urban100	0.6509	0.6965	0.7081	**0.7128**

TABLE VI

Average Results of PSNR(dB) on the Classic5 and LIVE1 Dataset. The Best Result for Each Dataset Is **Highlighted**

Dataset \ Method	JPEG	ARCNN[11]	Zhang's[30]	ARN
		QF = 10		
Classic5	27.82	28.87	28.89	**29.28**
LIVE1	27.98	29.03	28.82	**29.27**
		QF = 20		
Classic5	30.12	30.82	30.91	**31.25**
LIVE1	30.28	31.05	30.86	**31.34**
		QF = 30		
Classic5	31.48	31.97	32.06	**32.27**
LIVE1	31.60	32.16	31.98	**32.39**
		QF = 40		
Classic5	32.43	32.68	32.81	**32.99**
LIVE1	32.53	32.91	32.81	**33.13**

from testing sets, as seen in Fig. 11, 12, and 13. Viewing the whole pictures, obviously, our ARN could recover better details and sharper edges. Specially, in the green windows in *Butterfly*, *ppt3*, and *Woman*, we can see more clear patterns, music notations, and lips in images recovered by our ARN. In a word, based on the adaptive structure of the proposed ARN, our method can recover better images with high visual quality.

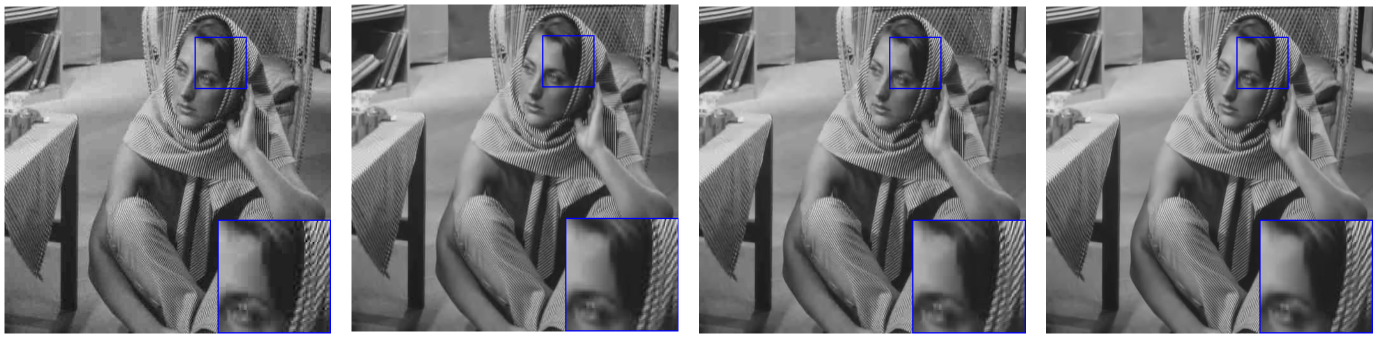

JPEG /28.34 /0.8525 Zhang's method /29.66 /0.8715 ARCNN /29.53 /0.8671 ARN **/30.26 /0.8852**

Fig. 15. Deblocking comparison on *Barbara* with state-of-the-art methods (QF = 20). (Zoom in for better view.)

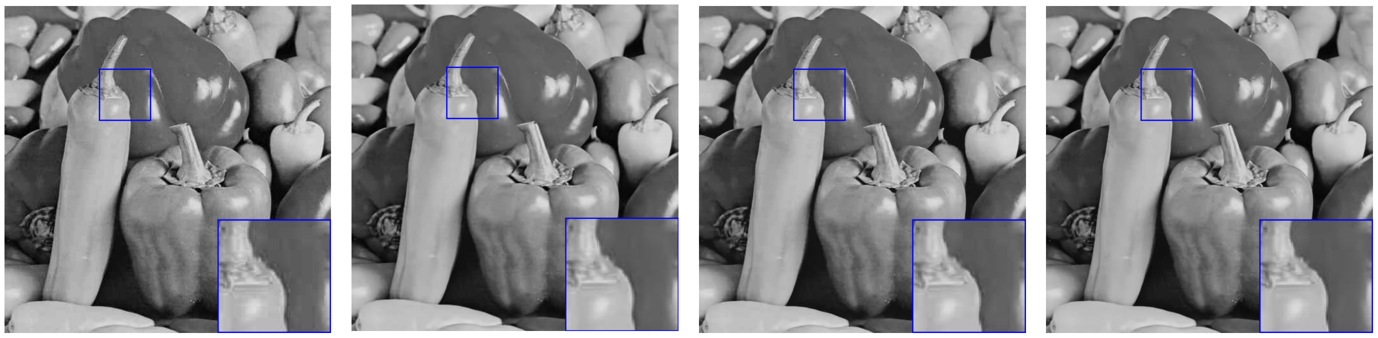

JPEG /33.68 /0.8582 Zhang's method /34.12 /0.8654 ARCNN /33.89 /0.8616 ARN **/34.25 /0.8673**

Fig. 16. Deblocking comparison on *Peppers* with state-of-the-art methods (QF = 30). (Zoom in for better view.)

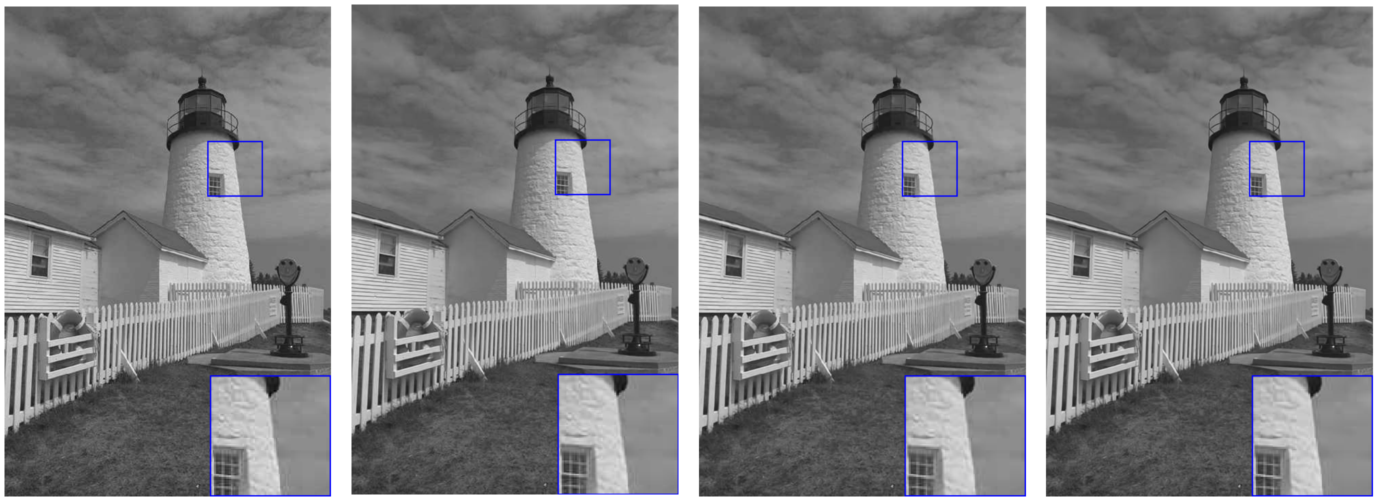

JPEG /32.88 /0.8924 Zhang's method /33.20 /0.8939 ARCNN /33.25 /0.8934 ARN **/33.56 /0.8976**

Fig. 17. Deblocking comparison on *Lighthouse3* with state-of-the-art methods (QF = 40). (Zoom in for better view.)

C. JPEG Image Deblocking

In order to further demonstrate the applicability of our proposed ARN, we investigate the JPEG image deblocking problem. It aims to suppress the block artifacts in the JPEG compressed images. Since ARCNN [11] and Zhang's method [34] are both classic methods, it would be persuasive to compare our method with them. We distorted the images by JPEG compression and set four compression quality factor (QF): 10, 20, 30, and 40 for the JPEG encoder.

1) Training Details: Following [11], we use Berkeley Segmentation Dataset BSD500 [22] as the training set and evaluate all the methods on the standard classic5 and LIVE1 datasets. Similarly, we split these training images into overlapping patches of size 50×50 with stride of 10.

2) Quantitative Results: From Table VI and Table VII, we can find that our ARN has superiority than other methods in both PSNR and SSIM. To be more specific, the average results of our ARN exceeds the second best from 0.18dB to 0.39dB. Likewise, the average SSIM values also have absolute advantage.

3) Visual Quality: To further study the deblocking results, we took a random sample of four images as QF = 10 (Fig. 14), 20 (Fig. 15), 30 (Fig. 16), and 40 (Fig. 17) respectively.

TABLE VII

AVERAGE RESULTS OF SSIM ON THE CLASSIC5 AND LIVE1 DATASET. THE BEST RESULT FOR EACH DATASET IS **HIGHLIGHTED**

Dataset \ Method	JPEG	ARCNN[11]	Zhang's[30]	ARN
	QF = 10			
Classic5	0.7595	0.7868	0.7885	**0.8004**
LIVE1	0.7722	0.7994	0.7960	**0.8077**
	QF = 20			
Classic5	0.8344	0.8439	0.8464	**0.8542**
LIVE1	0.8502	0.8614	0.8602	**0.8696**
	QF = 30			
Classic5	0.8666	0.8715	0.8716	**0.8775**
LIVE1	0.8838	0.8902	0.8873	**0.8958**
	QF = 40			
Classic5	0.8849	0.8862	0.8877	**0.8900**
LIVE1	0.9027	0.9055	0.9050	**0.9095**

As shown in the blue windows of the four images, we can draw the same conclusions that our ARN reconstructs better results as before. In our recovered images, it is hard to find any artificial blocks, which could be seen in the results by other methods.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed adaptive residual networks (ARN) for high-quality image restoration. It is made up of six cascaded adaptive shortcuts, convolutional layers and PReLUs. Each adaptive shortcut contains two small convolutional layers, followed by PReLU activation layers and one adaptive skip connection. Various scaling parameters are assigned to the different inputs of the shortcuts, and the scaling is considered as part of the weight parameters of the ARN model, which can be trained adaptively according to different applications. It is noteworthy that the adaptive shortcut helps on recovering clean image details and tackling the gradient vanishing problem. Moreover, we illustrate some effective strategies to train our ARN, including PReLU, xiaver initialization and adam optimization. Finally, extensive experimental results validate the excellent performance of our ARN.

In the future, we would like to deeply investigate the property of different image restoration problems via our ARN. By analyzing and comparing similarities and differences of various restoration applications, we expect to find more rules and strategies when reconstructing high-quality images.

REFERENCES

[1] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. Washington, DC, USA: Winston, 1977.

[2] V. Agarwal, A. V. Gribok, and M. A. Abidi, "Image restoration using l1 norm penalty function," *Inverse Problems Sci. Eng.*, vol. 15, no. 8, pp. 785–809, 2007.

[3] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. Ser. B (Methodol.)*, vol. 58, pp. 267–288, Jan. 1996.

[4] X. Zhang, M. Burger, X. Bresson, and S. Osher, "Bregmanized nonlocal regularization for deconvolution and sparse reconstruction," *SIAM J. Imag. Sci.*, vol. 3, no. 3, pp. 253–276, 2010.

[5] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 1838–1857, Jul. 2011.

[6] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[7] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[8] W. Dong, G. Shi, and X. Li, "Nonlocal image restoration with bilateral variance estimation: A low-rank approach," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 700–711, Feb. 2013.

[9] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. CVPR*, Jun. 2014, pp. 2862–2869.

[10] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. ECCV*, 2014, pp. 184–199.

[11] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. ICCV*, 2015, pp. 576–584.

[12] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. ICML*, 2008, pp. 1096–1103.

[13] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. CVPR*, Jun. 2012, pp. 2392–2399.

[14] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, and Q. Dai, "Supervised hash coding with deep neural network for environment perception of intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 284–295, Jan. 2018.

[15] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. (2016). "Densely connected convolutional networks." [Online]. Available: https://arxiv.org/abs/1608.06993

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[18] K. He, X. Zhang, S. Ren, and J. Sun. (2016). "Identity mappings in deep residual networks." [Online]. Available: https://arxiv.org/abs/1603.05027

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[20] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*, 2015, pp. 1026–1034.

[22] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. ICCV*, 2001, pp. 416–423.

[23] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 1–10.

[24] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. CVPR*, 2005, pp. 60–65.

[25] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[26] H. Liu, R. Xiong, J. Zhang, and W. Gao, "Image denoising via adaptive soft-thresholding based on non-local samples," in *Proc. CVPR*, Jun. 2015, pp. 484–492.

[27] F. Chen, L. Zhang, and H. Yu, "External patch prior guided internal clustering for image denoising," in *Proc. ICCV*, 2015, pp. 603–611.

[28] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. ICCV*, 2013, pp. 1920–1927.

[29] C.-Y. Yang and M.-H. Yang, "Fast direct super-resolution by simple functions," in *Proc. ICCV*, 2013, pp. 561–568.

[30] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Proc. ACCV*, 2014, pp. 111–126.

[31] D. Dai, R. Timofte, and L. Van Gool, "Jointly optimized regressors for image super-resolution," *Eurographics*, vol. 34, no. 2, pp. 95–104, May 2015.

[32] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1532–1546, Sep. 2000.

[33] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images," *IEEE Trans. Image Process.*, vol. 16, no. 5, pp. 1395–1411, May 2007.

[34] X. Zhang, R. Xiong, X. Fan, S. Ma, and W. Gao, "Compression artifact reduction by overlapped-block transform coefficient estimation with block similarity," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4613–4626, Dec. 2013.

[35] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. ICCV*, 2011, pp. 479–486.

[36] J. Xu, L. Zhang, W. Zuo, D. Zhang, and X. Feng, "Patch group based nonlocal self-similarity prior learning for image denoising," in *Proc. ICCV*, 2015, pp. 244–252.

[37] Z. Yongbing, Z. Debin, Z. Jian, X. Ruiqin, and G. Wen, "Interpolation dependent image downsampling," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3291–3296, Nov. 2011.

[38] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Proc. NIPS*, 2009, pp. 769–776.

[39] K. Lee, D. S. Kim, and T. Kim, "Regression-based prediction for blocking artifact reduction in JPEG-compressed images," *IEEE Trans. Image Process.*, vol. 14, no. 1, pp. 36–48, Jan. 2005.

[40] S. Liu and A. C. Bovik, "Efficient DCT-domain blind measurement and reduction of blocking artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1139–1149, Dec. 2002.

[41] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.

[42] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[43] Y. Wu, H. Zhao, and L. Zhang, "Image denoising with rectified linear units," in *Proc. NIPS*, 2014, pp. 142–149.

[44] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[45] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, 2013, pp. 1139–1147.

[48] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[49] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surfaces*, 2010, pp. 711–730.

[50] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. CVPR*, Jun. 2015, pp. 5197–5206.

Yongbing Zhang received the B.A. degree in english and the M.S. and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2004, 2006, and 2010, respectively. He joined the Graduate School at Shenzhen, Tsinghua University, Shenzhen, China, in 2010, where he is currently an Associate Professor. He received the Best Student Paper Award at the IEEE International Conference on Visual Communication and Image Processing in 2015.

His current research interests include video processing, image and video coding, video streaming, and transmission.

Lulu Sun received the B.E. degree in electrical engineering and automation from the China University of Mining and Technology, Beijing, China, in 2015. She is currently pursuing the M.E. degree with Tsinghua University, Beijing.

Her research interests include image processing, computer vision, and deep learning.

Chenggang Yan received the B.S. degree in computer science from Shandong University in 2008 and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2013. He was an Assistant Research Fellow with Tsinghua University. He is currently a Professor with Hanzhou Dianzi Univeristy.

His research interests include intelligent information processing, machine learning, image processing, computational biology, and computational photography. He has authored or co-authored over 30 refereed journal and conference papers. As a co-author, he received the best paper awards at the International Conference on Game Theory for Networks 2014 and the SPIE/COS Photonics Asia Conference 9273 2014, and the Best Paper Candidate at the International Conference on Multimedia and Expo 2011.E.

Xiangyang Ji received the B.S. degree in materials science and the M.S. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1999 and 2001, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He joined Tsinghua University, Beijing, in 2008, where he is currently a Professor with the Department of Automation.

His current research interests cover signal processing, image/video processing, and machine learning.

Qionghai Dai received the M.S. and Ph.D. degrees in computer science and automation from Northeastern University, Shenyang, China, in 1994 and 1996, respectively. He is currently a Professor with the Department of Automation and the Director of the Broadband Networks and Digital Media Laboratory, Tsinghua University, Beijing. He has authored or co-authored over 200 conference and journal papers and two books.

His research interests include computational photography and microscopy, computer vision and graphics, and intelligent signal processing. He is an Associate Editor of JVCI, the IEEE TNNLS, and the IEEE TIP.