Image Super-Resolution via Dual-State Recurrent Networks

Wei Han[1]*, Shiyu Chang[2]*, Ding Liu[1]†, Mo Yu[2], Michael Witbrock[2], Thomas S. Huang[1]†
[1]University of Illinois at Urbana-Champaign [2]IBM Research
{weihan3, dingliu2, t-huang1}@illinois.edu,
shiyu.chang@ibm.com, {yum,witbrock}@us.ibm.com

Abstract

Advances in image super-resolution (SR) have recently benefited significantly from rapid developments in deep neural networks. Inspired by these recent discoveries, we note that many state-of-the-art deep SR architectures can be reformulated as a single-state recurrent neural network (RNN) with finite unfoldings. In this paper, we explore new structures for SR based on this compact RNN view, leading us to a dual-state design, the Dual-State Recurrent Network (DSRN). Compared to its single-state counterparts that operate at a fixed spatial resolution, DSRN exploits both low-resolution (LR) and high-resolution (HR) signals jointly. Recurrent signals are exchanged between these states in both directions (both LR to HR and HR to LR) via delayed feedback. Extensive quantitative and qualitative evaluations on benchmark datasets and on a recent challenge demonstrate that the proposed DSRN performs favorably against state-of-the-art algorithms in terms of both memory consumption and predictive accuracy. The code for our method is publicly available[1].

1. Introduction

In the problem of single-image super-resolution (SR), the aim is to recover a high-resolution (HR) image from a single low-resolution (LR) image. In recent years, SR performance has been significantly improved due to rapid developments in deep neural networks (DNNs). Specifically, convolutional neural networks (CNNs) and residual learning [16] have been widely applied in much recent SR work [10, 12, 20, 21, 22, 25, 35, 38].

In these approaches, two principles have been consistently observed. The first is that increasing the depth of a CNN model improves SR performance; a deeper model with more parameters can represent a more complex mapping from LR to HR images. In addition, increasing network depth enlarges the size of receptive fields, providing more contextual information that can be exploited to reconstruct missing HR components. The second principle is that adding residual connections (globally [20], locally [21] or jointly [35]) prevents the problems of vanishing and exploding gradients, facilitating the training of deep models.

While these recent models have demonstrated promising results, there are also drawbacks. One major issue is that increasing the depth of models by adding new layers introduces more parameters, and thus raises the likelihood of model overfitting. At the same time, larger models demand more storage space, which is a hurdle to deployment in resource-constrained environments (*e.g.* mobile systems). To resolve this issue, the Deep Recursive Residual Network (DRRN) [35] inspired by the Deeply-Recursive Convolutional Network (DRCN) [21] shares weights across different residual units and achieves state-of-the-art performance with a small number of parameters.

Separate efforts [6, 24, 39] in neural architectural design have recently shown that commonly-used deep structures can be represented more compactly using recurrent neural networks (RNNs). Specifically, Liao and Poggio [24] demonstrated that a weight-sharing Residual Neural Network (ResNet) [16] is equivalent to a shallow RNN. Inspired by their findings, we first explore the connections between the neural architectures of existing SR algorithms and their compact RNN formulations. We note that previous SR models with recursive computation and weight sharing, including DRRN and DRCN, work at a single spatial resolution (bicubic interpolation is first applied to upscale LR images to a desired spatial resolution). This enables their model structures to be represented as a unified single-state RNN. Thus, both DRRN and DRCN can be viewed as a finite unfolding in time of the same RNN structure, but with different transition functions. This is illustrated in Figure 1, and will be discussed in detail in Section 3. It is worth mentioning that we follow the terminology used in [24], where a "state" can be considered as corresponding to a "layer" in the normal RNN setting.

*Authors contributed equally to this work
†Ding Liu and Thomas Huang's research works are supported in part by US Army Research Office grant W911NF-15-1-0317.
[1]https://github.com/weihan3/dsrn

Based on this compact RNN view of state-of-the-art SR models, in this paper we explore new structures to extend the frontier of SR. The first approach in improving a conventional RNN model is generally to make it multi-layer. We apply this experience in designing the SR architecture in our compact RNN view by adding an additional state, rendering our model a Dual-State Recurrent Network (DSRN), where the two states operate at different spatial resolutions. Specifically, the bottom state captures information at LR, while the top state operates in the HR regime. As with a conventional two-layer stacked RNN, there is a connection from the bottom to the top state via deconvolutional operations. This provides information flow from LR to HR at every single unrolling time. In addition, to allow information flow from previously predicted HR features to LR features, we incorporate a delayed feedback mechanism [8] from the top (HR) state to the bottom one. The overall structure of the proposed DSRN is shown in Figure 2, which not only utilizes parameters efficiently but also allows both LR and HR signals to contribute jointly to learning the mappings.

To demonstrate the effectiveness of the proposed method, we compare DSRN with other recent image SR approaches on four common benchmarks [4, 19, 30, 46] as well as on the DIV2K dataset from the "New Trends in Image Restoration and Enhancement workshop and challenge on image super-resolution (NTIRE SR 2017)" [1]. Extensive experimental results validate that DSRN delivers higher parameter efficiency, low memory consumption and high restoration accuracy.

2. Related Work

Single image SR has been widely studied in the past few decades and has an extensive literature. In recent years, due to the fast development of deep learning, significant progress has been made in this field. Dong *et al.* [10] first exploited a fully convolutional neural network, termed SRCNN, to predict the nonlinear LR-HR mapping. It demonstrated superior performance to many other example-based learning paradigms, such as nearest neighbor [13], sparse representation [44, 43], neighborhood embedding [5, 37], random forest [32], *etc.* Although all layers of a SRCNN are trained jointly in an end-to-end fashion, conceptually the network is split into three stages: patch representation, non-linear mapping, and reconstruction.

Much of the later work follows a similar network design with more complicated building blocks or advanced optimization techniques [34, 28, 11, 27]. Wang *et al.* [41] proposed a sparse coding network (SCN) that encodes a sparse representation prior for image SR and can be trained end-to-end, demonstrating the benefit of domain expertise in sparse coding for image SR. Both external and self examples were utilized to synthesize the HR prediction via a neural network in [42].

Inspired by the success of very deep models [16] on ImageNet challenges [9], Kim *et al.* [20] proposed a very deep CNN, VDSR, which stacks 20 convolutional layers with 3×3 kernels. Both residual learning and adjustable gradient clipping are used to prevent vanishing and exploding gradients. However, as the model gets deeper, the number of parameters increases. To control the size of the model, DRCN introduces 16 recursive layers, each with the same structure and shared parameters. Moreover, DRCN makes use of skip connections and recursive supervision to mitigate the difficulty of training. Tai *et al.* [35] discovered that many residual SR learning algorithms are based on either global residual learning or local residual learning, which are insufficient for very deep models. Instead, they proposed the DRRN that applies both global and local learning while remaining parameter efficient via recursive learning. More recently, Tong *et al.* [38] proposed making use of Densely Connected Networks (DenseNet) [17] instead of ResNet as the building block for image SR. They demonstrated that the DenseNet structure is better at combining features at different levels, which boosts SR performance.

Apart from deep models working on bicubic upscaled input images, Shi *et al.* [34] used a compact network model to conduct convolutions on LR images directly and learned upscaling filters in the last layer, which considerably reduces the computation cost. Similarly, Dong *et al.* [11] adopted deconvolution layers to accelerate SRCNN in combination with smaller filter sizes and more convolution layers. However, these networks are relatively small and have difficulty capturing complicated mappings owing to limited network capacity. The Laplacian Pyramid Super-Resolution Network (LapSRN) [22] works on LR images directly and progressively predicts sub-band residuals on various scales. Lim *et al.* [25] proposed the Enhanced Deep Super-Resolution (EDSR) network and a multi-scale variant, which learns different scaled mapping functions in parallel via weight sharing.

It is noteworthy that most SR algorithms minimize the mean squared reconstruction error (*i.e.* via ℓ_2 loss). They often suffer from regression-to-the-mean due to the ill-posed nature of single image SR, resulting in blurry predictions and poor subjective scores. To overcome this drawback, Generative Adversarial Networks have been used along with perceptual loss for SR [23, 31]. Subjective evaluation by mean-opinion-score showed huge improvement over other regression-based methods.

Our work is also strongly related to and built upon the idea of viewing a ResNet as an unrolled RNN. It was first proposed in [24], which aids understanding of a family of deep structures from the perspective of RNNs. Later, Chen et al. [45] unified several different residual functions to provide a better understanding of the design of DNNs with high learning capacity. Recently, the equivalence to RNNs has been further extended to DenseNet. Based on this finding, Dual Path Networks [6] were proposed and showed superior performance to DenseNet and ResNet in a varity of applications.

3. Single-State Recurrent Networks

In this section, we first revisit the discovery that a ResNet with shared weights can be reformulated as a recurrent system. Then, based on this view, we unite the recent development of SR models with such RNN reformulations to show DRCN and DRRN are structurally equivalent to an unrolled single-state RNN.

To establish the equivalence, we adopt the commonly used definition of a RNN, which is characterized by a set of states and transition functions among the states. A RNN often consists of the input state, output state, and the recurrent states. Depending on the number recurrent states, we describe RNNs as "single-state" (*i.e.* one recurrent state) or "dual-state" (*i.e.* two recurrent states). An illustration of a single-state RNN is shown in Figure 1(a). The input, output, and recurrent states are represented as x, y and s respectively. The arrow link indicates the state transition function. The square on the directed cycle indicates that the recurrent function travels one time step forward during the unfolding. Interested readers are referred to [47] for detailed information on this general formulation of a RNN.

Based on Figure 1(a), we unfold along the temporal direction to a fixed length T. The unfolded graph is shown in figure 1(b), and the dynamics of a single-state RNN can be characterized by:

$$\begin{aligned} s^t &= f_{\text{input}}(x^t) + f_{\text{recurrent}}(s^{t-1}) \\ y^t &= f_{\text{output}}(s^t), \end{aligned} \tag{1}$$

where the upper script t indicates the t-th unrolling. The parameters of f_{input}, f_{output}, and $f_{\text{recurrent}}$ are often time-independent, which means these parameters are reused at every unfolding step. This allows us to unify ResNet, DRCN, and DRRN as unrolled networks with the same recurrent structure but with the different realizations of $f_{\text{recurrent}}$ and different rules of parameter sharing.

ResNet: We consider a ResNet in its simplest form without any down-sampling or up-sampling operations. In other words, both of the spatial dimensions and feature dimensions remain the same across all intermediate layers. To render Figure 1(b) equivalent to a ResNet with T residual blocks, one possible technique is to make:

- s^0 be the input image I or a function of I.
- $x^t = 0, \forall t \in \{1, \dots T\}$, and $f_{\text{input}}(0) = 0$. Thus, the state transition becomes $s^t = f_{\text{recurrent}}(s^{t-1})$.
- The recurrent function $f_{\text{recurrent}}$ be the same as a conventional residual block, which contains two convolutional layers with skip connections as shown in Figure 1(c). Differences in color indicate different sets of parameters.
- The prediction state y^t be calculated only at the time T as the final output.

It is worth mentioning that the only difference between an unrolled RNN following the above definitions and a conventional ResNet is that the parameters in $f_{\text{recurrent}}$ need to be reused among all residual blocks.

DRCN: To realize the DRCN expressible by the same single-state RNN, we define s^0 and x^t in the same way as for the ResNet. Since DRCN recursively applies only a single convolutional layer to the input feature map 16 times, with the parameters of the layer reused across the whole network, we could use a single convolutional layer to express $f_{\text{recurrent}}$. The graph is illustrated in Figure 1(d). Moreover, unlike the ResNet where the output is predicted only at the end of unfolding, DRCN utilizes recursive supervision, which generates an output y^t at every unfolding t. The final HR prediction of DRCN is the weighted sum of the outputs at every unfolding t.

DRRN: The recurrent structure of DRRN differs only slightly from a ResNet. In a ResNet, the skip connection comes from the previous residual block, whereas in a DRRN the skip connection always comes from the first unrolled state s_0. Figure 1(e) shows the equivalent recurrent function for a DRRN with one recursive block (*i.e.* $B = 1$) using the definition in the original paper.

4. Dual-State Recurrent Networks

Drawing on the connections between state-of-the-art SR models and RNNs, we have investigated new compact RNN architectures for image SR. Specifically, we propose a dual-state design, which adopts two recurrent states enable use of features from both LR and HR spaces. The RNN view of our DSRN is shown in Figure 2(a) and is introduced as follows.

Dual-state design: Unlike single-state models working at the same spatial resolution, DSRN incorporates information from both the LR and HR spaces. Specifically, s_l and s_h in Figure 2(a) indicate the LR state and HR state, respectively. Four colored arrows indicate the transition functions between these two states. The blue (f_{lr}), orange (f_{hr}) and yellow (f_{up}) links exist in a conventional two-layer RNN, providing information flow from LR to LR, HR to HR, and LR to HR, respectively. To further enable two-way information flows between s_l and s_h, we add the green link, which is inspired by the delayed feedback mechanism of traditional

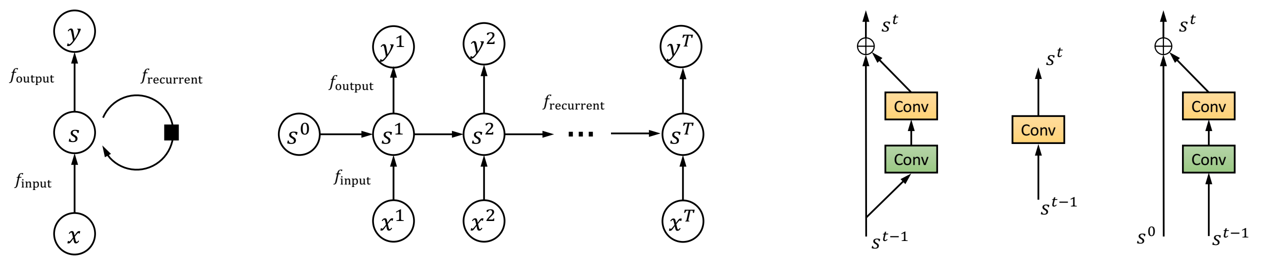

Figure 1. (a) An example of a single-state RNN, which is characterized by an input state x, output state y and a single recurrent state s. The arrow links indicate the state transition function. The black square represents the state transition function delayed for one time step. (b) Finite unfolding (T times) of a single-state RNN. (c) - (e) The required recurrent function to make a single-state RNN equivalent to ResNet, DRCN, and DRRN, respectively. Different colors of the "Conv" layers indicate different parameters.

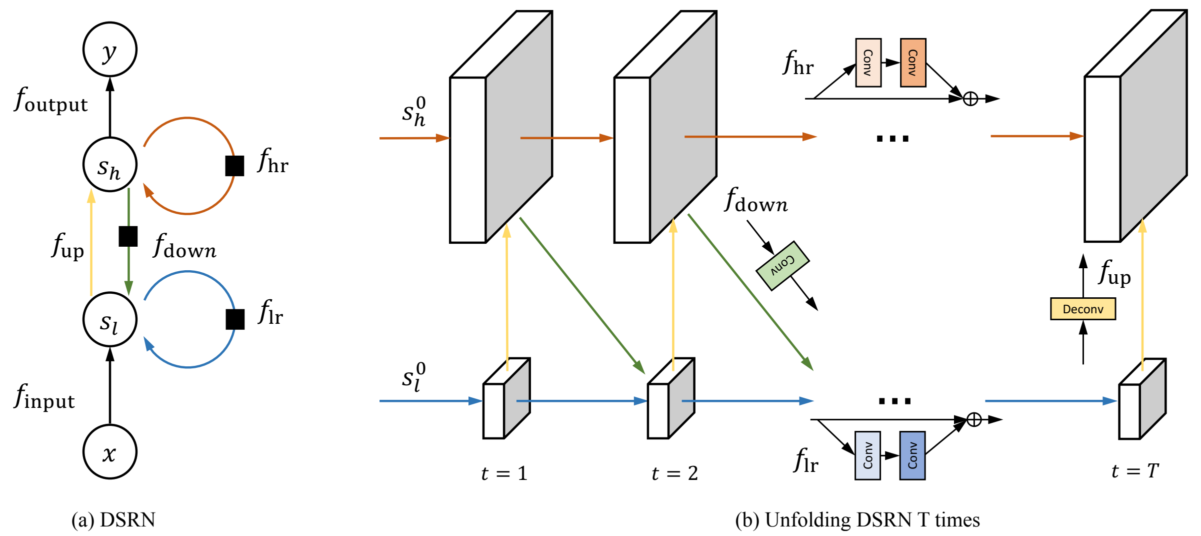

Figure 2. (a) The recurrent representation of the proposed DSRN, whose graph definition is the same as Figure 1(a). (b) The unrolled DSRN. Edges with the same color have identical state transition functions and shared parameters. The structures of four specific transition functions have been illustrated correspondingly. "Conv" blocks with different colors indicate different parameters.

multi-layer RNNs. Here, it introduces a delayed HR to LR connection. The overall dynamics of our DSRN is given as:

$$\begin{aligned} s_h^t &= f_{\text{up}}(s_l^t) + f_{\text{hr}}(s_h^{t-1}), \text{ and} \\ s_l^t &= f_{\text{input}}(x^t) + f_{\text{lr}}(s_l^{t-1}) + f_{\text{down}}(s_h^{t-1}). \end{aligned} \tag{2}$$

Figure 2(b) demonstrates the same concept via an unfolded graph, where the top row represents HR state while the bottom one is LR. This design choice encourages feature specialization for different resolutions and information sharing across different resolutions.

Transition functions: Our model is characterized by six transition functions. f_{up}, f_{down}, f_{lr}, and f_{hr} as illustrated in Figure 2(b). Specifically, we use the standard residual block for both self-transitions. A single convolutional layer is used for the down-sampling transition and a single transposed convolutional (or deconvolutional) layer is used for the up-sampling transition. The strides in both inter-state layers are set to be the same as the SR upscaling factor.

Unfolding details: Similarly to unfolding a single-state RNN to obtain a ResNet, for image SR, we let x^t have no contribution to calculating the state transition. In other words,

$$f_{\text{input}}(x^t) = 0, \tag{3}$$

for any choice of x^t (*e.g.* choose $x^t = 0$, $\forall t$). Furthermore, we set s_l^0 as the output of two convolutional layers with skip connections, which takes the LR input image and transform it into a desired feature space. In addition, s_h^0 is set to zero. Finally, we use deep supervision for the HR prediction, as discussed below.

Deep supervision: The unrolled DSRN is capable of making a prediction at every time step t. Denote

$$\hat{y}^t = f_{\text{output}}(s_h^t) \tag{4}$$

as a prediction at the t^{th} unfolding, where f_{output} is characterized by a single convolutional layer. Then, instead of taking the prediction only at the final unfolding T, we average all the predictions as

$$\hat{I}_h = \frac{1}{T}\sum_{t=1}^{T}\hat{y}^t. \tag{5}$$

Thus, every unrolled layer directly connects to the loss layer to facilitate the training of such a very deep network. Moreover, the model predicts the residual image and minimizes the following mean square error

$$\mathscr{L}(\hat{I}_h, I_h) = \frac{1}{2}||\hat{I}_h - r_i||^2, \tag{6}$$

where I_h is the group-truth image in HR and $r_i = I_h - bicubic(I_l)$ is the residual map between the ground truth and bicubic upsampled LR image.

5. Experiments

In this section, we first provide implementation details, including both model hyper-parameters and training data augmentation. Then we analyze a number of design choices and their contributions to final performance. Finally, we compare DSRN to other state-of-the-art methods on several benchmark datasets.

5.1. Datasets

To evaluate the proposed DSRN algorithm, we train our model using 91 images proposed in [44] and test on the following datasets: Set5 [4], Set14 [46], B100 [30] and Urban100 [19]. The training data is augmented in a similar way to previous methods [20, 35], which includes 1) random flipping along the vertical or horizontal axis; 2) random rotation by 90°, 180° or 270°; and 3) random scaling by a factor from [0.5, 0.6, 0.7, 0.8, 0.9, 1]. Tensorflow is used for our full data processing pipeline; the LR training images are generated by the built-in bicubic down-sampling function. We additionally test our algorithm on the DIV2K dataset of the NTIRE SR 2017 challenge [1], where we use the provided training and validation sets with all of the aforementioned data augmentations except random scaling.

5.2. Implementation Details

We use our model to super-resolve only the luminance channel of images, and use bicubic interpolation to upscale the other two color channels, following [20, 21, 35]. We train independent models for each scale (×2, ×3, and ×4) with 64 filters on the first input convolutional layer and 128 filters in the rest of the network. All layers use 3 × 3 convolution filters. Due to our dual-state design, the feature maps of s_l and s_h in each time step have the same spatial dimensions as the LR and HR images, respectively. We zero-pad the boundaries of feature maps to ensure the spatial size of each feature map is the same as the input size after the convolution is applied.

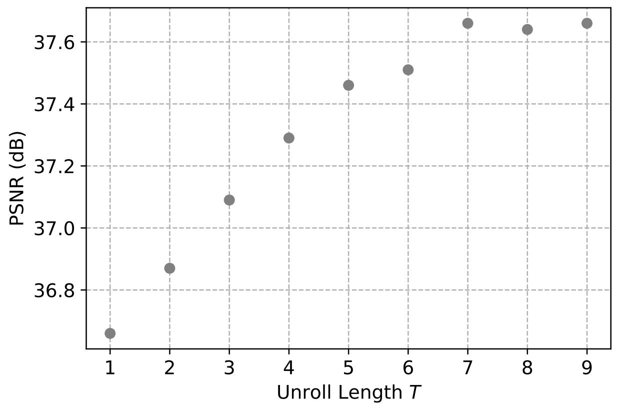

Figure 3. Unrolling length v.s. PSNR performance of our DSRN with ×2 upscaling on Set5 dataset.

All the weights in the network are initialized with a uniform distribution using the method proposed in [14]. We use standard stochastic gradient descent (SGD) with momentum 0.95 as our optimizer to minimize the MSE loss function in Equation (6). We search for the best initial learning rate from $\{0.1, 0.03, 0.01\}$ and reduce it by a factor of 10 three times during the entire training process. This learning rate annealing is driven by observing that the loss on the validation set stops decreasing. Gradient clipping at 0.5 is adopted during training to prevent the gradient explosion. We sample image patches with a size of 128 × 128 and use a mini-batch size of 16 to train our network.

We observe that the recursion defined in Equation (2) may lead to an exponential increase in the scale of feature values, especially when T is large. In [24], the authors proposed the use of unshared batch normalization at every unfolding time to resolve this issue. Batch normalization is not used in our network; we found that normalizing the scale with two scalar parameters was sufficient. Specifically, we use one unshared PReLU [15] activation for each recurrent state after every unrolling step. All other layers have ordinary ReLU as the activation function.

5.3. Model Analysis

In this section, we analyze our proposed model in the following respects:

Unrolling length: The unrolling length T changes the maximum effective depth of the unrolled network. In particular, for a DSRN with T times unrolling, the maximum number of convolution layers between input and output of the network is $2T + 4$. The multiplier 2 comes from the two layers in a residual block, while the extra 4 is from the auxiliary input and output layers. However, the number of model

Table 1. Quantitative evaluation of state-of-the-art SR algorithms: average PSNR/SSIM/IFC for scale factors $\times 2$, $\times 3$ and $\times 4$. **Bold red** text indicates the best and underlined blue text the second best performance.

Algorithm	Scale	Set5	Set14	BSDS100	Urban100
		PSNR / SSIM / IFC	PSNR / SSIM / IFC	PSNR / SSIM / IFC	PSNR / SSIM / IFC
Bicubic	2	33.65 / 0.930 / 6.166	30.34 / 0.870 / 6.126	29.56 / 0.844 / 5.695	26.88 / 0.841 / 6.319
A+ [37]	2	36.54 / 0.954 / 8.715	32.40 / 0.906 / 8.201	31.22 / 0.887 / 7.464	29.23 / 0.894 / 8.440
SRCNN [10]	2	36.65 / 0.954 / 8.165	32.29 / 0.903 / 7.829	31.36 / 0.888 / 7.242	29.52 / 0.895 / 8.092
FSRCNN [11]	2	36.99 / 0.955 / 8.200	32.73 / 0.909 / 7.843	31.51 / 0.891 / 7.180	29.87 / 0.901 / 8.131
SelfExSR [19]	2	36.49 / 0.954 / 8.391	32.44 / 0.906 / 8.014	31.18 / 0.886 / 7.239	29.54 / 0.897 / 8.414
RFL [32]	2	36.55 / 0.954 / 8.006	32.36 / 0.905 / 7.684	31.16 / 0.885 / 6.930	29.13 / 0.891 / 7.840
SCN [41]	2	36.52 / 0.953 / 7.358	32.42 / 0.904 / 7.085	31.24 / 0.884 / 6.500	29.50 / 0.896 / 7.324
VDSR [20]	2	37.53 / 0.958 / 8.190	32.97 / 0.913 / 7.878	31.90 / 0.896 / 7.169	30.77 / 0.914 / 8.270
DRCN [21]	2	37.63 / 0.959 / 8.326	32.98 / 0.913 / 8.025	31.85 / 0.894 / 7.220	30.76 / 0.913 / 8.527
LapSRN [22]	2	37.52 / 0.959 / **9.010**	33.08 / 0.913 / **8.505**	31.80 / 0.895 / **7.715**	30.41 / 0.910 / 8.907
DRRN [35]	2	**37.74** / 0.959 / 8.671	**33.23** / **0.914** / 8.320	32.05 / 0.897 / N.A.	**31.23** / **0.919** / **8.917**
DSRN	2	37.66 / **0.959** / 8.585	33.15 / 0.913 / 8.169	**32.10** / **0.897** / 7.541	30.97 / 0.916 / 8.598
Bicubic	3	30.39 / 0.868 / 3.596	27.64 / 0.776 / 3.491	27.21 / 0.740 / 3.168	24.46 / 0.736 / 3.661
A+ [37]	3	32.60 / 0.908 / 4.979	29.24 / 0.821 / 4.545	28.30 / 0.784 / 4.028	26.05 / 0.798 / 4.883
SRCNN [10]	3	32.76 / 0.908 / 4.682	29.41 / 0.823 / 4.373	28.41 / 0.787 / 3.879	26.24 / 0.800 / 4.630
FSRCNN [11]	3	33.15 / 0.913 / 4.971	29.53 / 0.826 / 4.569	28.52 / 0.790 / 4.061	26.42 / 0.807 / 4.878
SelfExSR [19]	3	32.63 / 0.908 / 4.911	29.33 / 0.823 / 4.505	28.29 / 0.785 / 3.922	26.45 / 0.809 / 4.988
RFL [32]	3	32.45 / 0.905 / 4.956	29.15 / 0.819 / 4.532	28.22 / 0.782 / 4.023	25.87 / 0.791 / 4.781
SCN [41]	3	32.60 / 0.907 / 4.321	29.24 / 0.819 / 4.006	28.32 / 0.782 / 3.553	26.21 / 0.801 / 4.253
VDSR [20]	3	33.66 / 0.921 / 5.088	29.77 / 0.834 / 4.606	28.83 / 0.798 / 4.043	27.14 / 0.829 / 5.045
DRCN [21]	3	33.82 / 0.922 / 5.202	29.76 / 0.833 / 4.686	28.80 / 0.797 / **4.070**	27.15 / 0.828 / 5.187
LapSRN [22]	3	33.78 / 0.921 / 5.194	29.87 / 0.833 / 4.665	28.81 / 0.797 / 4.057	27.06 / 0.827 / 5.168
DRRN [35]	3	**34.03** / **0.924** / **5.397**	29.96 / 0.835 / 4.878	**28.95** / **0.800** / N.A.	**27.53** / **0.838** / **5.456**
DSRN	3	33.88 / 0.922 / 5.221	**30.26** / **0.837** / **4.892**	28.81 / 0.797 / 4.051	27.16 / 0.828 / 5.172
Bicubic	4	28.42 / 0.810 / 2.337	26.10 / 0.704 / 2.246	25.96 / 0.669 / 1.993	23.15 / 0.659 / 2.386
A+ [37]	4	30.30 / 0.859 / 3.260	27.43 / 0.752 / 2.961	26.82 / 0.710 / 2.564	24.34 / 0.720 / 3.218
SRCNN [10]	4	30.49 / 0.862 / 2.997	27.61 / 0.754 / 2.767	26.91 / 0.712 / 2.412	24.53 / 0.724 / 2.992
FSRCNN [11]	4	30.71 / 0.865 / 2.994	27.70 / 0.756 / 2.723	26.97 / 0.714 / 2.370	24.61 / 0.727 / 2.916
SelfExSR [19]	4	30.33 / 0.861 / 3.249	27.54 / 0.756 / 2.952	26.84 / 0.712 / 2.512	24.82 / 0.740 / 3.381
RFL [32]	4	30.15 / 0.853 / 3.135	27.33 / 0.748 / 2.853	26.75 / 0.707 / 2.455	24.20 / 0.711 / 3.000
SCN [41]	4	30.39 / 0.862 / 2.911	27.48 / 0.751 / 2.651	26.87 / 0.710 / 2.309	24.52 / 0.725 / 2.861
VDSR [20]	4	31.35 / 0.882 / 3.496	28.03 / 0.770 / 3.071	27.29 / 0.726 / 2.627	25.18 / 0.753 / 3.405
DRCN [21]	4	31.53 / 0.884 / 3.502	28.04 / 0.770 / 3.066	27.24 / 0.724 / 2.587	25.14 / 0.752 / 3.412
LapSRN [22]	4	31.54 / 0.885 / 3.559	28.19 / 0.772 / 3.147	27.32 / 0.728 / **2.677**	25.21 / 0.756 / 3.530
DRRN [35]	4	**31.68** / **0.889** / **3.703**	**28.21** / **0.772** / **3.252**	**27.38** / **0.728** / N.A.	**25.44** / **0.764** / **3.676**
DSRN	4	31.40 / 0.883 / 3.500	28.07 / 0.770 / 3.147	27.25 / 0.724 / 2.599	25.08 / 0.747 / 3.297

parameters remains independent of the length of unrolling. Essentially, T controls the trade-off between model capacity and computation cost. We study the influence of T by training the model with different unrolling lengths. The empirical results are shown in Figure 3. The test performance increases when the number of unfolding steps increases, but the benefit seems to diminish after $T = 7$. Unless otherwise mentioned, we use $T = 7$ for all our models. It is worth mentioning that we also experimented with stochastic depth [18] by randomly sampling T during training, but we observed no improvement in validation accuracy.

Parameter sharing: We empirically find parameter sharing to be crucial for training a deep recursive model. As shown in Table 2, the same model with untied weights performs much more poorly than its weight-sharing counterpart. Specifically, we observe around 0.2dB performance drop across all three upscaling scales when changing from shared weights to untied weights. We speculate that the model with untied weights suffers a larger risk of model over-fitting and much slower training convergence, both of which diminish the model's restoration accuracy.

Dual-state and delayed feedback: We compare our DSRN

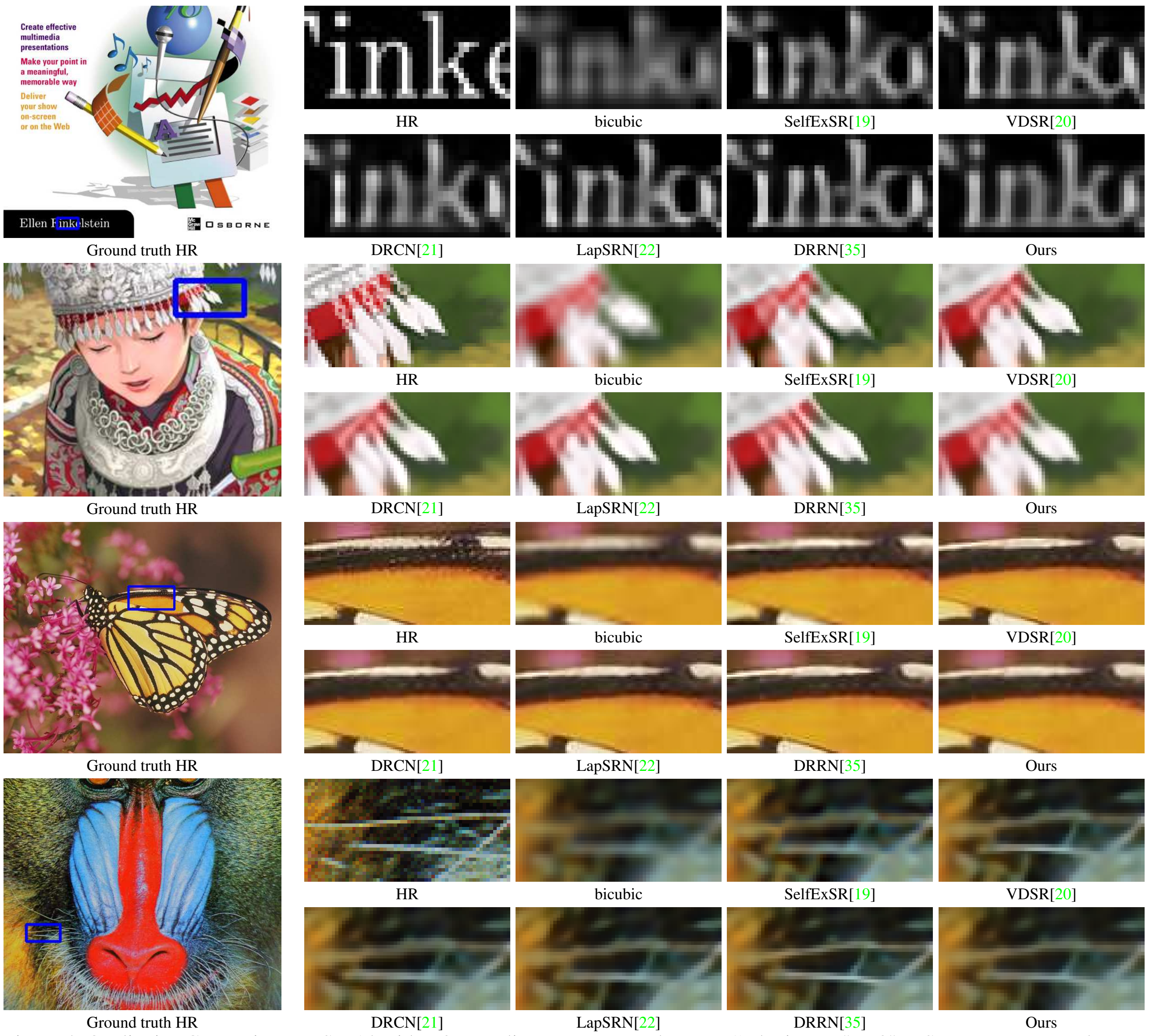

Figure 4. Qualitative Comparison on Set 14 with $\times 3$ upscaling. From top to bottom: 1) the image "ppt3", DSRN reconstructs sharp text with less artifacts. 2) the image "comic". 3) the image "monarch", DSRN finds less blurry dots along the edge of the wing. 4) the image "baboon".

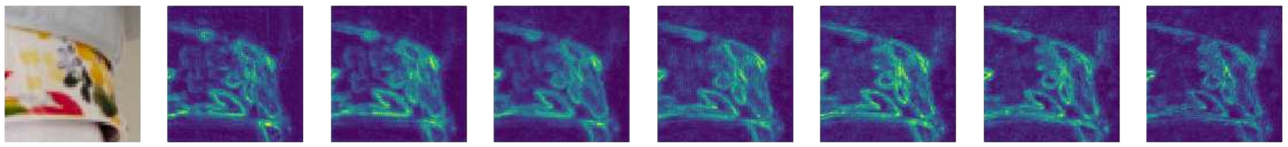

Figure 5. Feature visualization: input image patch and the energy maps of the output at HR states (7 unrolled timestamps in total).

with two baselines under the same unrolling time steps to understand how each module of our model contributes to the final performance: 1) a single-state RNN unrolled ResNet; and 2) a dual-state RNN without delayed feedback connections. The quantitative comparison on the NTIRE SR 2017 challenge is shown in Table 2. Comparing the single-state baseline and the DSRN without feedback, it is clear that considering information from both LR and HR spaces as two separated states provides performance gains. In addition, comparing our models with and without feedback, we

Table 2. Quantitative evaluation (in PSNR) of the proposed DSRN, its variants, and other state-of-the-art SR algorithms on track 1 of the NTIRE SR 2017 challenge. **Bold red** text indicates the best and underlined blue text indicates the second best performance. The number in () indicates ranking in the challenge.

	Method	x2	x3	x4
Ours	Single-state baseline	34.66	30.80	28.80
	DSRN w/o parameter sharing	34.71	30.85	28.81
	DSRN w/o delayed feedback	34.89	30.95	28.99
	DSRN	**34.96**	31.12	29.03
Others	EDSR+ [25] (1)	34.93	**31.13**	**29.04**
	Wang *et al.* [36] (2)	34.47	30.77	28.82
	Bae *et al.* [3] (3)	34.66	30.83	28.83
	SelNet [7] (4)	34.29	30.52	28.55
	BTSRN [12] (5)	34.19	30.44	28.49

realize that incorporating such an information flow from HR space back to LR space consistently improves performance on all three different scales. In all, both the dual-state and delayed feedback designs are beneficial to our model.

State visualization Since DSRN has independent scaling parameters on each unrolled state, the model implicitly learns a weighted-average of all the unrolled states for the final prediction. Empirically we observe that this strategy performs better than output from the last state only. To demonstrate how the network aggregates different unrolled states, we show feature response maps at different unrolling steps in Figure 6, demonstrating that the network distributes slightly different features to each unrolled state.

5.4. Comparison with the State-of-the-Art

We provide results of evaluation of our model on several public benchmark datasets in Table 1, with three commonly-used evaluation metrics: Peak Signal-to-Noise Ratio (PSNR), Structural SIMilarity (SSIM) [40] and the Information Fidelity Criterion (IFC) [33]. Specifically, we perform a comprehensive comparison between our method and 10 other existing SR algorithms, including both deep learning and non-deep-learning based methods. Note that many recent deep learning based competitors, including VDSR, LapSRN and DRRN, use 291 training samples with the additional 200 from the training set of Berkeley Segmentation Dataset [2], while our model was trained on only the 91 images. Still, our DSRN method achieves competitive performance across all datasets and scales. It achieves particularly strong performance in the $\times 2$ and $\times 3$ settings.

In addition, we report quantitative evaluations on the recently developed DIV2K dataset and comparisons with top-ranking algorithms in Table 2. Our method achieves competitive performance with the best algorithm, EDSR+[25], and outperforms all the other algorithms by a large margin, which demonstrates the effectiveness of our proposed dual-state recurrent structure.

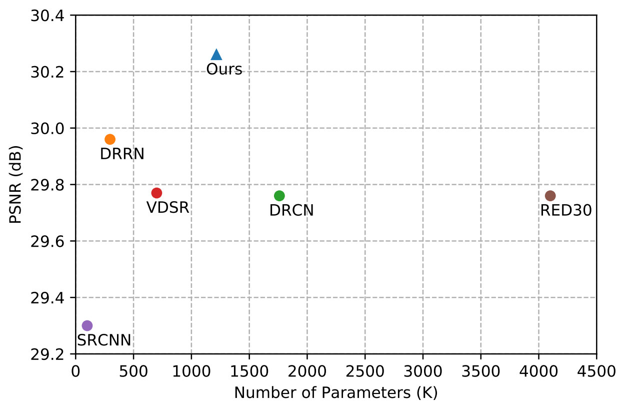

Figure 6. Comparison of the PSNR and the model size of recent SR methods for $\times 3$ upscaling on Set 14.

To further analyze the proposed DSRN against other state-of-the-art SR approaches in a qualitative manner, in Figure 4 we present several visual examples of super-resolved images on Set14 with $x3$ upscaling among different SR approaches. For these competing methods, we use SR results publicly released by the authors. As shown in Figure 4, our method can construct sharp and detailed structures and is less prone to generating spurious artifacts.

Furthermore, the proposed DSRN benefits from inherent parameter sharing and therefore obtains higher parameter efficiency compared to other methods. In Figure 6, we illustrate the parameters-to-PSNR relationship of our model and several state-of-the-art methods, including SRCNN, VDSR, DRCN, DRRN and RED30 [29]. Our method represents a favorable trade-off between model size and SR performance, and has modest inference time. The DSRN takes 0.4s on the x4 task with a 288x288 output image size, on an NVIDIA Titan X GPU.

6. Conclusion

In this work, we have provided a unique formulation that expresses many state-of-the-art SR models as a finite unfolding of a single-state RNN with various recurrent functions. Based on this, we extend existing methods by considering a dual-state design; the two hidden states of our proposed DSRN operate at different spatial resolutions. One captures the LR information while the other one targets the HR domains. To ensure two-way communication between states, we integrate a delayed feedback mechanism. Thus, the predicted features from both LR and HR states can be exploited jointly for final predictions. Extensive experiments on benchmark datasets have demonstrated that the proposed DSRN performs favorably against state-of-the-art SR models in terms of both efficiency and accuracy. For the future work, we will explore use of our proposed DSRN to capture temporal dependencies for video SR [26].

References

[1] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017. 2, 5

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011. 8

[3] W. Bae, J. Yoo, and J. C. Ye. Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification. *arXiv preprint arXiv:1611.06345*, 2016. 8

[4] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 2, 5

[5] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *CVPR*, volume 1, pages I–I. IEEE, 2004. 2

[6] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017. 1, 3

[7] J.-S. Choi and M. Kim. A deep convolutional neural network with selection units for super-resolution. In *CVPRW*, pages 1150–1156. IEEE, 2017. 8

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. In *ICML*, 2015. 2

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. 2

[10] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014. 1, 2, 6

[11] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016. 2, 6

[12] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, Z. Wang, X. Wang, and T. S. Huang. Balanced two-stage residual networks for image super-resolution. In *CVPR Workshops*, 2017. 1, 8

[13] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002. 2

[14] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 5

[15] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 5

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[17] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016. 2

[18] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661. Springer, 2016. 6

[19] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 2, 5, 6, 7

[20] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 1, 2, 5, 6, 7

[21] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016. 1, 5, 6, 7

[22] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 1, 2, 6, 7

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 2

[24] Q. Liao and T. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016. 1, 2, 5

[25] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 1, 2, 8

[26] D. Liu, Z. Wang, Y. Fan, X. Liu, Z. Wang, S. Chang, and T. Huang. Robust video super-resolution with learned temporal dynamics. In *ICCV*, pages 2507–2515, 2017. 8

[27] D. Liu, Z. Wang, N. Nasrabadi, and T. Huang. Learning a mixture of deep networks for single image super-resolution. In *ACCV*, pages 145–156. Springer, 2016. 2

[28] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, and T. S. Huang. Robust single image super-resolution via deep networks with sparse prior. *TIP*, 25(7):3194–3207, 2016. 2

[29] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, pages 2802–2810, 2016. 8

[30] 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 *ICCV*, 2001. 2, 5

[31] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *ICCV*, Oct 2017. 2

[32] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *CVPR*, 2015. 2, 6

[33] H. R. Sheikh, A. C. Bovik, and G. De Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on image processing*, 14(12):2117–2128, 2005. 8

[34] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016. 2

[35] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017. 1, 2, 5, 6, 7

[36] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1110–1121. IEEE, 2017. 8

[37] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014. 2, 6

[38] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *ICCV*, 2017. 1, 2

[39] A. Veit, M. J. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, 2016. 1

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 8

[41] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *ICCV*, 2015. 2, 6

[42] Z. Wang, Y. Yang, Z. Wang, S. Chang, W. Han, J. Yang, and T. Huang. Self-tuned deep super resolution. In *CVPRW*, pages 1–8, 2015. 2

[43] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *TIP*, 21(8):3467–3478, 2012. 2

[44] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *TIP*, 2010. 2, 5

[45] C. Yunpeng, J. Xiaojie, K. Bingyi, F. Jiashi, and Y. Shuicheng. Sharing residual units through collective tensor factorization in deep neural networks. *arXiv preprint arXiv:1703.02180*, 2017. 2

[46] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *ICCS*, 2010. 2, 5

[47] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. R. Salakhutdinov, and Y. Bengio. Architectural complexity measures of recurrent neural networks. In *NIPS*, 2016. 3