

Alma Mater Studiorum
University of Bologna

Artificial Intelligence
Machine Learning for Computer Vision
Alessandro Dicosola [Matr. 935563]

Super Resolution
Survey Any-Scale Deep Network (ASDN [1])

Contents

1	Introduction	1
2	Background	2
2.1	Deep Bi-Dense Network (DBDN[2])	2
2.1.1	Architectures	3
2.1.2	Results	4
2.2	Residual Channel Attention Network (RCAN[3])	6
2.3	Channel-wise and Spatial Feature Modulation Network (CSFM[4])	7
2.4	Deep Laplacian Pyramid Network (LapSRN[5]) and Multi-scale Deep Laplacian Pyramid Network (MS-LapSRN [6])	8
2.4.1	Features	8
2.4.2	Architecture	9
2.4.3	Loss	10
2.4.4	Depth	10
2.4.5	Results	10
2.5	Magnification-Arbitrary Network (MetaSR [7])	14
2.5.1	Background	14
2.5.2	Architecture	14
2.5.3	Results	16

1 Introduction

In computer vision the **super resolution** (hereinafter SR) task is an ill-posed problem for reconstructing a low resolution (hereinafter LR) image to an higher one (hereinafter SR - super resolution image); in order to do so an high resolution image is used (hereinafter HR).

Many method proposed trying to define the images in the features space and apply linear and non-linear operators in order to learn and reconstruct the high resolution image: using random forest, linear and non-linear regressor, manifold embedding[8].

Due to advancement of deep learning, SR task started to be achieved using deep neural networks: from the simplest one, SRCNN[9] with only three convolution for extracting and processing features used then for reconstructing the SR image to more complex ones that use short,long and dense skip connections [10][2] and recursive blocks [11][12], attention mechanism [3][4], meta-learning [7], multi-levels [5][6].

Studies done in deep learning has lead research to understand that deeper and wider networks performs better than shallow and narrow ones as well as using residual learning; skip connections (short,long and dense) are used for overcoming the exploding/vanishing gradient problem and at the same time they allow to let networks converge faster; dense connection allow also to reuse features from previous stages and attention mechanism allow networks to focus on

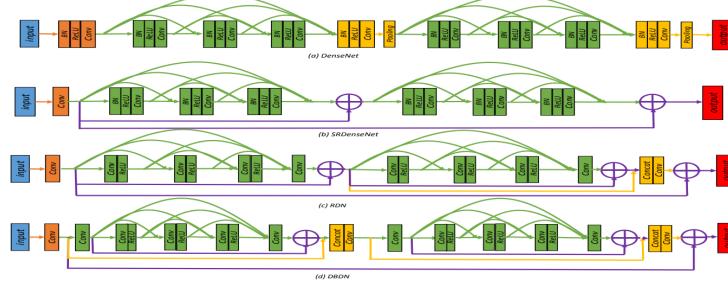


Fig. 2. Simplified structure of (a) DenseNet [26]. The green lines and layers denote the connections and layers in the dense block, and the yellow layers denote the transition and pooling layer. (b) SRDenseNet [15]. The green layers are the same dense block structures as those in DenseNet. The purple lines and element-wise \oplus refer to the skip connection. (c) RDN [13]. The yellow lines denote concatenating the blocks output for reconstruction and the green layers are residual dense blocks. (d) DBDN. The yellow lines and layers denote the inter-block dense connection and the green layers are the intra dense blocks. The output goes into upsampling reconstruction layers. The orange layers after input are the feature extraction layers in all models.

Figure 1: Comparison of different network that are using dense connection.

most important features (channel-wise and spatial-wise).

SR task achieved with deep learning trying to learn a mapping between LR to HR images given a specific scale. The studies done up to now always used fixed integer scales ($2x, 3x, 4x, \dots$) but none had explored decimal scales but Meta-SR[7] (using meta-learning).

Therefore **Any-Scale Deep Network** (ASDN) tried to explore decimal scale in SR task learning the mapping in an end-to-end way.

2 Background

Here will be explored the most important networks referenced by ASDN [1].

2.1 Deep Bi-Dense Network (DBDN[2])

Skip connections allow to highlight one of the most important hyperparameter in deep network: the depth.

But in SR task is important also to use the features learned at the beginning of the network: therefore dense connection were adopted, moreover new architecture arose such as *SRDenseNet* and *RDN*.

All the previous mentioned architectures have a problem: the feature reusing is not really achieved as we can see from the Figure 1 (the concatenation in the last row use features computed before the sequence of convolutions).

Therefore the aim of Deep Bi-Dense Network (DBDN) is to be able to reuse the feature in the network thanks to dense connections used along the network and within each block.

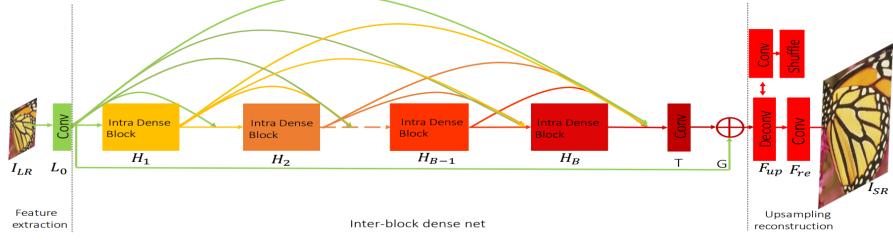


Fig. 3. Network structure

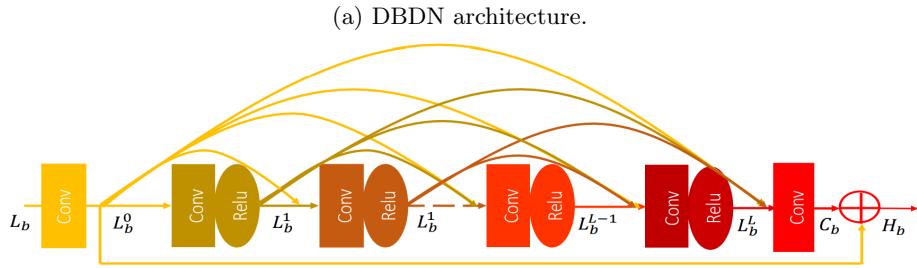


Fig. 4. Intra dense blocks

(b) Intra dense block.

2.1.1 Architectures

We can see from Figure 2a how features are reused: the features extracted by the single feature extraction convolution are processed by a sequence of **Intra dense block**.

The *Intra dense block* contains at the beginning and at the end two convolutions which compress the input and the output channels (to n_r) and in the middle L convolutions (with n_g filters) and ReLUs, connected densely, process the compressed input. The compressions are necessary due to the concatenation at the beginning of each intra dense block (Figure 2a) and at the end of each intra dense block (Figure 2b):

- $n_r * \text{number of IDBs before}$.
- $n_r + L * n_g$ at the end of each IDB.

The local residual learning allow the flowing of the gradients allowing a faster convergence.

Moreover a global skip connections is used in order to use low-frequency information from the LR image and allow the gradient to flow from the end to the beginning of the network.

The SR image is reconstructed using either deconvolution or sub-pixel convolution[13]

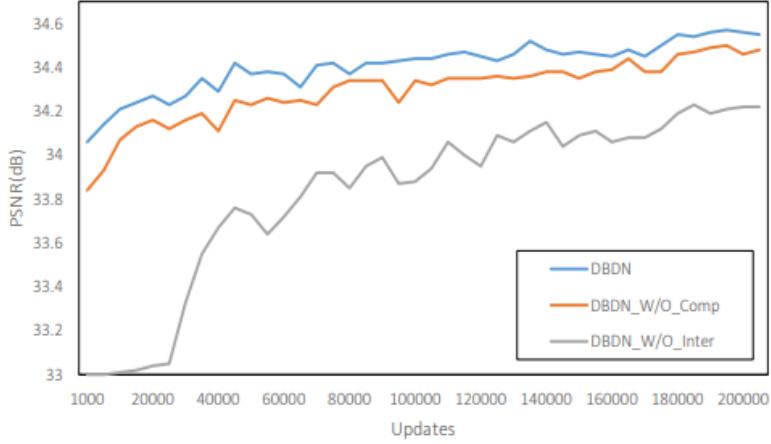


Fig. 8. Discussion about transition layer, skip connection and inter-block dense connectivity in DBDN. The results are evaluated with Set5 for $3\times$ enlargement in 200 epochs

Figure 3: Ablation studies on DBDN.

2.1.2 Results

Ablation studies on compression layer and inter-block dense connection From the Figure 3 we can see that *Intra dense-block connection* and *compression layers* are important for achieving good results.

Quantitative and qualitative results Thanks to the feature reusing the model perform better than state-of-the-art network maintaining the number of parameters low.

Datasets	Scale	Bicubic	VDSR [7]	LapSRN [16]	DRRN [6]	SRDenseNet [15]	EDSR [14]	RDN [13]	DBPN [32]	DBDN(ours)	DBDN+(ours)
Set5	2×	33.66/0.9929	37.53/0.9587	37.52/0.9591	37.74/0.9591	-/-	38.11/0.9601	38.24/0.9614	38.09/0.9600	38.30/0.9617	38.35/0.9618
	3×	30.39/0.8682	33.66/0.9124	33.82/0.9227	34.03/0.9244	-/-	34.65/0.9282	34.71/0.9296	-/-	34.76/0.9299	34.83/0.9303
	4×	28.42/0.8104	31.35/0.8838	31.54/0.8855	31.68/0.8888	32.02/0.8934	32.46/0.8968	32.47/0.8990	32.47/0.8980	32.54/0.8991	32.70/0.9006
Set14	2×	30.24/0.8688	33.03/0.9124	33.08/0.9130	33.23/0.9136	-/-	33.92/0.9195	34.01/0.9212	33.85/0.9190	34.20/0.9224	34.34/0.9239
	3×	27.55/0.7742	29.28/0.8209	29.77/0.8314	29.96/0.8349	-/-	30.52/0.8462	30.57/0.8468	-/-	30.63/0.8478	30.75/0.8495
	4×	26.00/0.7027	28.01/0.7674	28.19/0.7720	28.21/0.7721	28.50/0.7782	28.80/0.7876	28.81/0.7871	28.82/0.7860	28.89/0.7890	29.00/0.7908
BSD100	2×	29.56/0.8431	31.90/0.8960	31.80/0.8950	32.05/0.8973	-/-	32.32/0.9013	32.34/0.9017	32.27/0.9000	32.39/0.9022	32.45/0.9028
	3×	27.21/0.7385	28.82/0.7976	28.82/0.7973	28.95/0.8004	-/-	29.25/0.8093	29.26/0.8093	-/-	29.31/0.8104	29.37/0.8112
	4×	25.96/0.6675	27.29/0.7251	27.32/0.7280	27.38/0.7284	27.53/0.7337	27.71/0.7420	27.72/0.7418	27.72/0.7400	27.76/0.7426	27.84/0.7446
Urban100	2×	26.88/0.8403	30.76/0.8946	30.41/0.9101	31.23/0.9188	-/-	32.93/0.9351	32.84/0.9347	32.51/0.9321	32.98/0.9364	33.36/0.9389
	3×	24.46/0.7349	26.24/0.7986	27.14/0.8272	27.53/0.8378	-/-	28.80/0.8653	28.79/0.8655	-/-	28.96/0.8682	29.17/0.8715
	4×	23.14/0.6577	25.18/0.7524	25.21/0.7553	25.44/0.7638	26.05/0.7819	26.64/0.8033	26.61/0.8028	26.38/0.7945	26.70/0.8050	27.00/0.8117
Manga109	2×	30.80/0.9339	37.16/0.9739	37.27/0.9740	37.60/0.9736	-/-	38.96/0.9769	39.18/0.9780	38.89/0.9775	39.46/0.9788	39.65/0.9793
	3×	26.95/0.8556	31.48/0.9317	32.19/0.9334	32.42/0.9359	-/-	34.17/0.9473	34.13/0.9484	-/-	34.46/0.9498	34.80/0.9512
	4×	24.89/0.7866	27.82/0.8856	29.09/0.8893	29.18/0.8914	27.83/0.8782	31.11/0.9148	31.00/0.9151	30.91/0.9137	31.23/0.9169	31.68/0.9198

Figure 4: Quantitative results on public benchmark. Red indicates best performance, blue indicates second best performance.

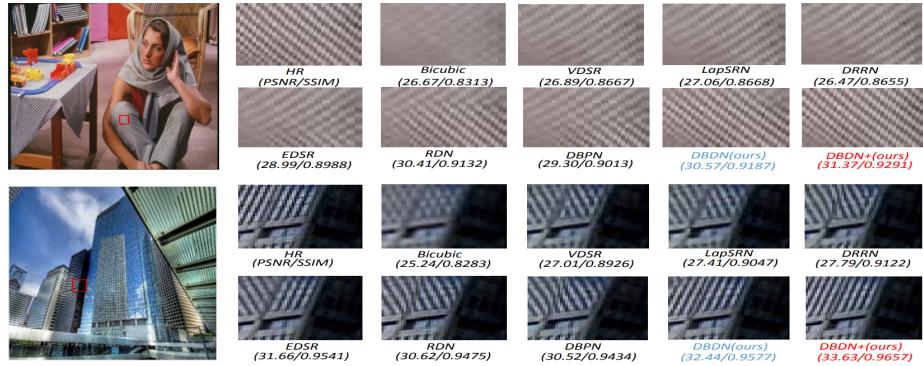


Figure 5: Qualitative results of DBDN.

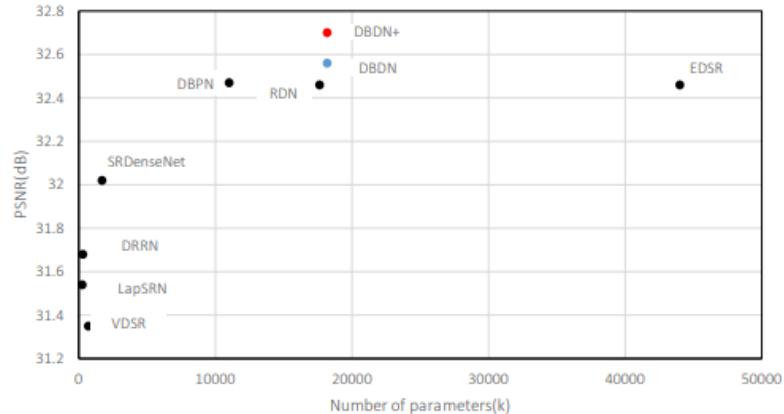


Fig. 7. Performance vs number of parameters. The results are evaluated with Set5 for 4 \times enlargement. Red indicates the best performance and blue indicates the second best.

Figure 6: Performance-Parameters study.

2.2 Residual Channel Attention Network (RCAN[3])

2.3 Channel-wise and Spatial Feature Modulation Network (CSFM[4])

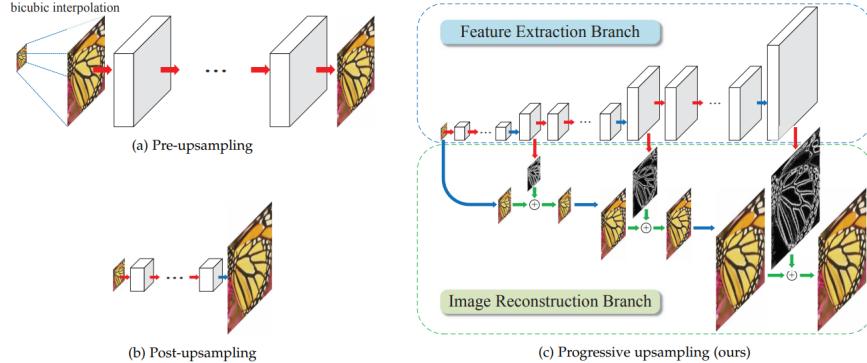


Fig. 1. **Comparisons of upsampling strategies in CNN-based SR algorithms.** Red arrows indicate **convolutional layers**, blue arrows indicate **transposed convolutions** (upsampling), and green arrows denote **elementwise addition** operators. (a) Pre-upsampling based approaches (e.g., SRCNN [9], VDSR [11], DRCN [12], DRRN [13]) typically use the bicubic interpolation to upscale LR input images to the target spatial resolution before applying deep networks for prediction and reconstruction. (b) Post-upsampling based methods directly extract features from LR input images and use sub-pixel convolution [14] or transposed convolution [15] for upsampling. (c) Progressive upsampling approach using the proposed Laplacian pyramid network reconstructs HR images in a coarse-to-fine manner.

Figure 7: LapSRN architecture.

2.4 Deep Laplacian Pyramid Network (LapSRN[5]) and Multi-scale Deep Laplacian Pyramid Network (MS-LapSRN [6])

2.4.1 Features

LapSRN in order to be able to reconstruct the SR image uses:

- **Charbonnier loss** because the L2 loss is not able to capture the underlying mapping of LR images to many HR images.
- **progressive reconstruction** of the SR image using the Laplacian Pyramid Framework [14].
- **residual learning**: learn the summation between the laplacian extracted by *features extraction branch* and the upscaled LR image in the *image reconstruction branch*

MS-LapSRN improve LapSRN introducing:

- **parameter sharing** across pyramid level and within pyramid levels in order to reduce the amount of parameters which increase with the scale (the greater the scale the deeper the network since there are more levels).
- **local skip connections** for avoiding the vanishing/exploding gradient problem with the increase in the depth of the network.
- **multi scale training**: the previous network was trained for each scale different networks.

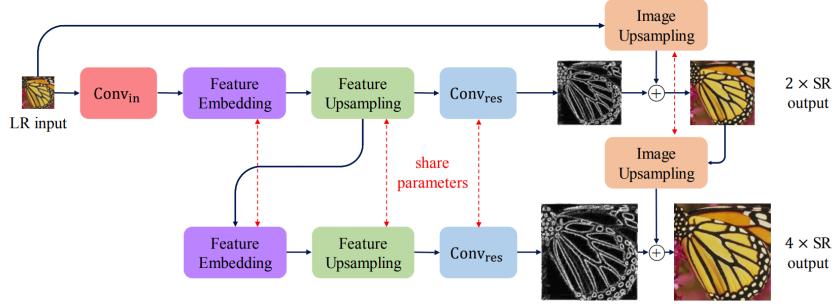


Fig. 3. **Detailed network architecture of the proposed LapSRN.** At each pyramid level, our model consists of a feature embedding sub-network for extracting non-linear features, transposed convolutional layers for upsampling feature maps and images, and a convolutional layer for predicting the sub-band residuals. As the network structure at each level is highly similar, we share the weights of those components across pyramid levels to reduce the number of network parameters.

Figure 8: MS-LapSRN architecture.

2.4.2 Architecture

In *LapSRN* [Figure 7] the **feature extraction branch** extract laplacian representations which are used for training (in an end-to-end fashion) the **image reconstruction branch** in order to reconstruct the SR image at the last level (due to the laplacian pyramid framework training on an higher scales lead to have also a network capable to resolve SR task for lower scale).

The residual learning allow the network to focus on high-frequency information (edges) instead of low-frequency ones.

In the *MS-LapSRN* [Figure 8] the feature extractor (*Conv_{in}*, Feature Embedding, Feature Upsampling, *Conv_{res}*) has always the same function: extract meaningful information from an input image in order to create a residual whose spatial dimension in 2x then the input one; therefore is logic to use same weights for doing so.

The *Feature embedding* has **R** recursive block [11] [12] which contains distinct **D** convolutional layers.

The structure of the recursive block [Figure 9b] use a skip connection directly connected to the input and a pre-activation inside the residual path [15].

Recursive block (or single convolution) allow to reduce the number of parameters and increase the depth of the network as well as the receptive field of the network in the last activation without increasing the memory footprint of the network.

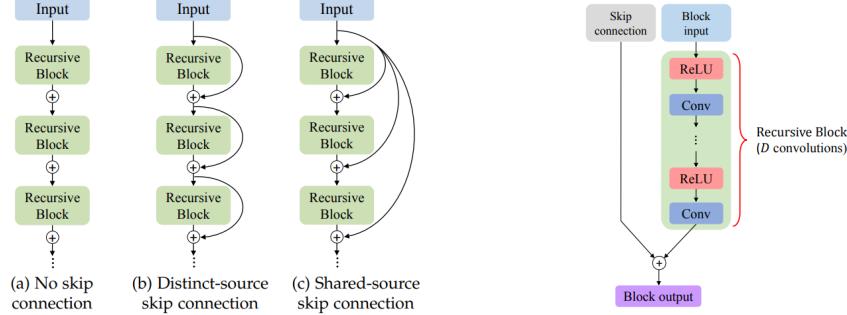


Fig. 4. **Local residual learning.** We explore three different ways of local skip connection in the feature embedding sub-network of the LapSRN for training deeper models.

(a) Differences between local skip connections.

Fig. 5. **Structure of our recursive block.** There are D convolutional layers in a recursive block. The weights of convolutional layers are distinct within the block but shared among all recursive blocks. We use the pre-activation structure [41] without the batch normalization layer.

(b) Recursive block used in MS-LapSRN.

2.4.3 Loss

Both *LapSRN* and *MS-LapSRN* use the **Charbonnier loss**: the former use it once the latter at each level; the equation is the following:

$$L_S = \frac{1}{N} \sum_{i=1}^N \sum_{l=1}^L \rho \times \left((y_l^{(i)} - x_l^{(i)}) - r_l^{(i)} \right)$$

where S is the target upsampling factor, $\rho = \sqrt{x^2 + \epsilon^2}$ (Charbonnier penalty function which is a differentiable L1 norm), x_l is the upscaled LR image at level l , y_l is the downsampled HR target image (y) at level l and r_l is the residual at level l .

The final loss is the sum of the loss for each level trained using images with different scales (**multi scales training**):

$$L = \sum_{s \in [2,4,8]} L_s$$

2.4.4 Depth

The depth of the network is: $(D \times R + 1) \times (L + 2)$ where D is the number of convolutions inside each recursive block, R is the number of recursive blocks, $+1$ represents the transposed convolution, $+2$ represent the convolutions that perform feature extraction and residual hallucination.

2.4.5 Results

Here will be presented only the results of **MS-LapSRN**[6] since it's an evolution of LapSRN.

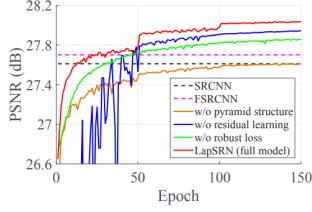


Fig. 6. **Convergence analysis.** We analyze the contributions of the pyramid structures, loss functions, and global residual learning by replacing each component with the one used in existing methods. Our full model converges faster and achieves better performance.

(a) Study on performance with different settings.

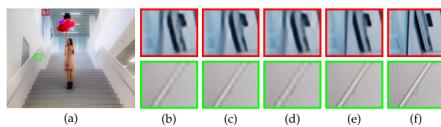


Fig. 7. **Contribution of different components in LapSRN.** (a) Ground truth HR image (b) without pyramid structure (c) without global residual learning (d) without robust loss (e) full model (f) HR patch.

(c) Contribution of different components in LapSRN.

TABLE 2
Ablation study of LapSRN. Our full model performs favorably against several variants of the LapSRN on both SET5 and SET14 for 4× SR.

GRL	Pyramid	Loss		SET5	SET14
✓	✓	Charbonnier	30.58	27.61	
✓	✓	Charbonnier	31.10	27.94	
✓	✓	\mathcal{L}_2	30.93	27.86	
✓	✓	Charbonnier	31.28	28.04	

(b) Ablation study on LapSRN.

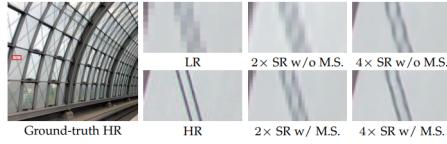


Fig. 8. **Contribution of multi-scale supervision (M.S.).** The multi-scale supervision guides the network training to progressively reconstruct the HR images and help reduce the spatial aliasing artifacts.

(d) Contribution of multi-scale supervision.

Figure 10: Performance studies done on MS-LapSRN

Ablation studies on pyramid structure, global residual learning and loss From the performance studies [Figure 10] it's possible to observe the contribution of each component used for the final result: the pyramid structures allow a faster convergence, the Charbonnier loss a better performance, the global residual learning both.

Studies on local skip connections From the local skip connection studies [Figure 11] we can see that the best is *shared source (SS)* which use as identity the same input.

Quantitative evaluation of local residual learning. We compare three different local residual learning methods on the URBAN100 dataset for 4× SR. Overall, the shared local skip connection method (LapSRN_{SS}) achieves superior performance for deeper models.

Model	Depth	LapSRN _{NS}	LapSRN _{DS}	LapSRN _{SS}
D5R2	24	25.16	25.22	25.23
D5R5	54	25.18	25.33	25.34
D5R8	84	25.26	25.33	25.38

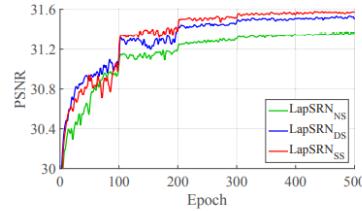


Fig. 9. **Comparisons of local residual learning.** We train our LapSRN-D5R5 model with three different local residual learning methods as described in Section 3.2.3 and evaluate on the SET5 for 4× SR.

Figure 11: Local skip connection studies done on MS-LapSRN

Parameter sharing in LapSRN. We reduce the number of network parameters by sharing the weights between pyramid levels and applying recursive layers in the feature embedding sub-network.

Model	#Parameters	BSDS100	URBAN100
LapSRN [16]	812k	27.32	25.21
LapSRN _{NS} -D10R1	407k	27.32	25.20
LapSRN _{NS} -D5R2	222k	27.30	25.16
LapSRN _{NS} -D2R5	112k	27.26	25.10

Quantitative evaluation of the number of recursive blocks R and the number of convolutional layers D in our feature embedding sub-network. We build LapSRN with different network depth by varying the values of D and R and evaluate on the BSDS100 and URBAN100 datasets for 4× SR.

Model	#Parameters	Depth	BSDS100	URBAN100
D2R5	112k	24	27.33	25.24
D2R12	112k	52	27.35	25.31
D2R20	112k	84	27.37	25.31
D4R3	185k	28	27.33	25.25
D4R6	185k	52	27.37	25.34
D4R10	185k	84	27.37	25.35
D5R2	222k	24	27.32	25.23
D5R5	222k	54	27.38	25.34
D5R8	222k	84	27.39	25.38
D10R1	407k	24	27.33	25.23
D10R2	407k	44	27.36	25.27
D10R4	407k	84	27.38	25.36

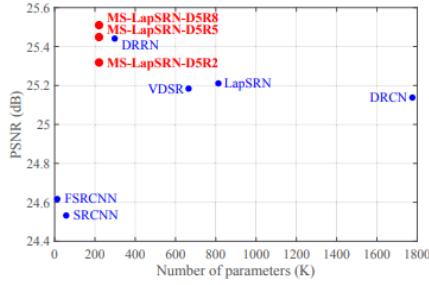


Fig. 15. **Number of network parameters versus performance.** The results are evaluated on the URBAN100 dataset for 4× SR. The proposed MS-LapSRN strides a balance between reconstruction accuracy and execution time.

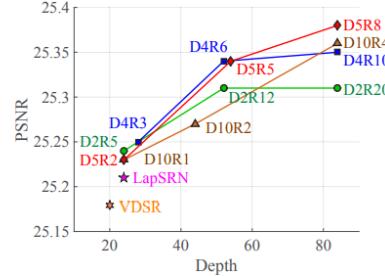


Fig. 10. **PSNR versus network depth.** We test the proposed model with different D and R on the URBAN100 dataset for 4× SR.

Figure 12: Studies on parameters (weights) and hyperparameters D and R on MS-LapSRN

Quantitative evaluation of state-of-the-art SR algorithms. We report the average PSNR/SSIM/MFCC for 2x, 4x, and 8x SR. Red and blue indicate the best and the second best performance, respectively. Both LapSRN [16] and the proposed MS-LapSRN do not reach any 3x SR images for training. To generate the results of 3x SR, we first generate a 2x SR image and then downsample it by 2x.

Algorithm	Scale	2x			4x			8x		
		PSNR	SSIM	MFCC	PSNR	SSIM	MFCC	PSNR	SSIM	MFCC
Biubic										
AI [1]	2x	33.00 / 0.931 / 6.38	32.25 / 0.879 / 6.12	29.27 / 0.844 / 5.69	26.89 / 0.831 / 5.219	30.96 / 0.963 / 6.214	—	—	—	—
RT [1]	2x	36.40 / 0.954 / 8.743	32.29 / 0.805 / 8.22	31.18 / 0.885 / 7.47	29.14 / 0.893 / 6.439	35.12 / 0.964 / 8.921	—	—	—	—
SrffSR [2]	2x	36.70 / 0.955 / 8.746	32.51 / 0.808 / 8.662	31.38 / 0.889 / 7.242	29.83 / 0.896 / 6.092	35.76 / 0.964 / 8.471	—	—	—	—
SRCCNN [9]	2x	36.70 / 0.955 / 8.746	32.51 / 0.808 / 8.662	31.38 / 0.889 / 7.242	29.83 / 0.896 / 6.092	35.76 / 0.964 / 8.471	—	—	—	—
FSRNN [10]	2x	36.40 / 0.954 / 7.388	32.19 / 0.805 / 7.085	31.26 / 0.885 / 6.903	29.82 / 0.897 / 7.234	35.31 / 0.962 / 7.260	—	—	—	—
SCN [10]	2x	36.40 / 0.954 / 7.388	32.19 / 0.805 / 7.085	31.26 / 0.885 / 6.903	29.82 / 0.897 / 7.234	35.31 / 0.962 / 7.260	—	—	—	—
VDSR [11]	2x	36.40 / 0.954 / 7.388	32.19 / 0.805 / 7.085	31.26 / 0.885 / 6.903	29.82 / 0.897 / 7.234	35.31 / 0.962 / 7.260	—	—	—	—
DRCN [12]	2x	37.45 / 0.950 / 8.334	32.06 / 0.812 / 8.023	31.85 / 0.895 / 7.228	30.76 / 0.943 / 8.527	37.63 / 0.976 / 9.541	—	—	—	—
LapSRN [16]	2x	37.45 / 0.950 / 8.334	32.06 / 0.812 / 8.023	31.85 / 0.895 / 7.228	30.76 / 0.943 / 8.527	37.63 / 0.976 / 9.541	—	—	—	—
MS-LapSRN-D982 (ours)	2x	37.45 / 0.950 / 8.334	32.06 / 0.812 / 8.023	31.85 / 0.895 / 7.228	30.76 / 0.943 / 8.527	37.63 / 0.976 / 9.541	—	—	—	—
MS-LapSRN-D983 (ours)	2x	37.72 / 0.960 / 8.346	32.42 / 0.814 / 8.226	32.09 / 0.904 / 7.206	31.71 / 0.942 / 8.534	37.71 / 0.975 / 9.570	—	—	—	—
MS-LapSRN-D984 (ours)	2x	37.72 / 0.960 / 8.346	32.42 / 0.814 / 8.226	32.09 / 0.904 / 7.206	31.71 / 0.942 / 8.534	37.71 / 0.975 / 9.570	—	—	—	—
Biubic										
AI [1]	4x	32.40 / 0.869 / 5.396	27.75 / 0.807 / 3.401	27.72 / 0.774 / 3.348	23.47 / 0.777 / 3.661	26.99 / 0.893 / 5.321	—	—	—	—
RT [1]	4x	32.40 / 0.869 / 5.397	29.15 / 0.820 / 4.343	28.31 / 0.785 / 4.028	26.85 / 0.799 / 4.883	29.93 / 0.912 / 4.880	—	—	—	—
SrffSR [2]	4x	32.40 / 0.869 / 5.397	29.15 / 0.820 / 4.343	28.31 / 0.785 / 4.028	26.85 / 0.799 / 4.883	29.93 / 0.912 / 4.880	—	—	—	—
SRCCNN [9]	4x	32.40 / 0.869 / 5.397	29.15 / 0.820 / 4.343	28.31 / 0.785 / 4.028	26.85 / 0.799 / 4.883	29.93 / 0.912 / 4.880	—	—	—	—
FSRNN [10]	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
SCN [10]	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
VDSR [11]	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
DRCN [12]	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
LapSRN [16]	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
MS-LapSRN-D982 (ours)	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
MS-LapSRN-D983 (ours)	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
MS-LapSRN-D984 (ours)	4x	33.10 / 0.914 / 5.397	29.27 / 0.820 / 4.506	28.33 / 0.791 / 4.061	26.43 / 0.800 / 4.878	31.10 / 0.921 / 4.912	—	—	—	—
Biubic										
AI [1]	8x	28.40 / 0.811 / 2.337	26.01 / 0.707 / 2.346	26.97 / 0.670 / 1.983	23.15 / 0.666 / 2.386	24.93 / 0.767 / 2.289	—	—	—	—
RT [1]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
SrffSR [2]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
SRCCNN [9]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
FSRNN [10]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
SCN [10]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
VDSR [11]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
DRCN [12]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
LapSRN [16]	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
MS-LapSRN-D982 (ours)	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
MS-LapSRN-D983 (ours)	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
MS-LapSRN-D984 (ours)	8x	30.17 / 0.853 / 3.295	22.24 / 0.747 / 2.924	22.17 / 0.706 / 2.538	21.25 / 0.722 / 3.331	26.80 / 0.841 / 3.055	—	—	—	—
Biubic										
AI [1]	2x	24.60 / 0.858 / 0.856	23.10 / 0.766 / 0.784	23.67 / 0.548 / 0.646	23.14 / 0.754 / 0.888	21.47 / 0.687 / 0.810	—	—	—	—
RT [1]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
SrffSR [2]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
SRCCNN [9]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
FSRNN [10]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
SCN [10]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
VDSR [11]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
DRCN [12]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
LapSRN [16]	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
MS-LapSRN-D982 (ours)	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
MS-LapSRN-D983 (ours)	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
MS-LapSRN-D984 (ours)	2x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
Biubic										
AI [1]	4x	24.60 / 0.858 / 0.856	23.10 / 0.766 / 0.784	23.67 / 0.548 / 0.646	23.14 / 0.754 / 0.888	21.47 / 0.687 / 0.810	—	—	—	—
RT [1]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
SrffSR [2]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
SRCCNN [9]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
FSRNN [10]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
SCN [10]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
VDSR [11]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
DRCN [12]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
LapSRN [16]	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
MS-LapSRN-D982 (ours)	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
MS-LapSRN-D983 (ours)	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
MS-LapSRN-D984 (ours)	4x	25.38 / 0.879 / 0.991	23.79 / 0.816 / 0.813	24.13 / 0.583 / 0.748	23.27 / 0.856 / 0.892	22.26 / 0.687 / 0.988	—	—	—	—
Biubic										
AI [1]	8x	24.60 / 0.858 / 0.856	23.10 / 0.766 / 0.784	23						

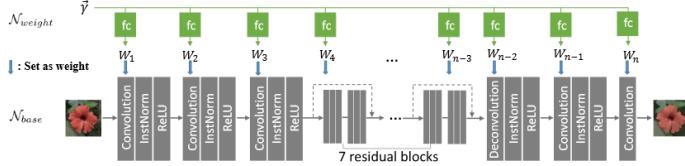


Figure 14: Parametrized Image Operator network.

2.5 Magnification-Arbitrary Network (MetaSR [7])

The limitation of SR task using fixed integer scales are:

- real application may be necessary upsampling on continues scales (such as in medical imaging, satellite imaging, ...).
- if the network is trained on a specific scale and is not capable to output intermediate scales (such as [5]) then there are also training cost and memory cost.

In order to overcame fixed integer problems [7] exploit **meta-learning** in order to be able to use decimal scales.

2.5.1 Background

The idea is used in [16] where research trained different parametrized operator (a generic function that transforms an image based on parameters) using a neural network exploiting meta-learning.

There are two networks Figure 14:

- N_{base} which applies the mapping
- $N_{weights}$ which learns the weights to use inside convolutions in N_{base} based on the parameters $\vec{\gamma}$

that are learned together end-to-end:

$$L = \|N_{base}(N_{weight}(\vec{\gamma}, I, E)) - f(I, \vec{\gamma})\|_2^2$$

2.5.2 Architecture

The **Feature Learning Module** extracts features at low resolution which are used by the **Meta Upscale Module** for generating the SR image.

The *Feature Learning Module* can be any state-of-the-art network while *Meta Upscale Module* replace the upsampling layer in the selected network.

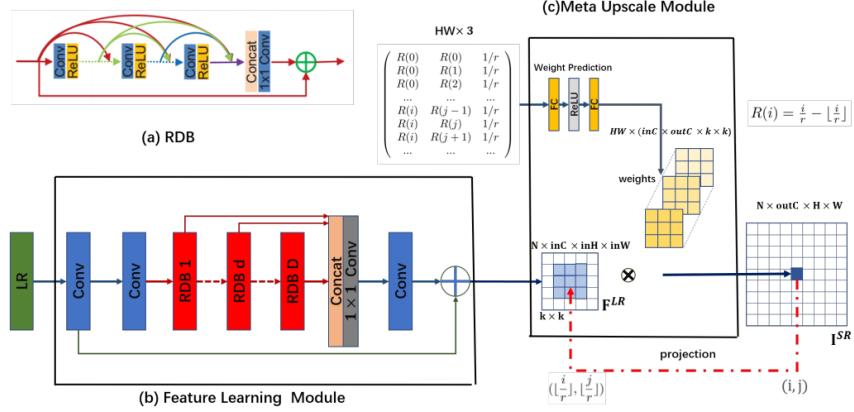


Figure 1. An instance of our Meta-SR based on RDN [36]. We also call the network Meta-RDN. (a) The Residual Dense Block proposed by RDN [36]. (b) The Feature Learning Module which generates the shared feature maps for arbitrary scale factor. (c) For each pixel on the SR image, we project it onto the LR image. The proposed Meta-Upscale Module takes a sequence of coordinate-related and scale-related vectors as input to predict the weights for convolution filters. By doing the convolution operation, our Meta-Upscale finally generate the HR image.

Figure 15: Meta-SR architecture.

Meta Upscale Module The **Meta Upscale Module** contains three different modules:

- **location projector** which project pixels from HR images to LR images: $(i', j') = T(i, j) = (\lfloor \frac{i}{r} \rfloor, \lfloor \frac{j}{r} \rfloor)$ where (i, j) are pixels in the HR image and (i', j') are pixels in the LR image.
- **weight prediction** which predicts the weights of the kernel for each pixel $W(i, j) = \phi(v_{ij}, \theta)$ where ϕ is a network with fully connected layers (2 with 256 hidden units ad ReLu as activation).

v_{ij} is a vector of pixel offset locations:

$$v_{ij} = (\frac{i}{r} - \lfloor \frac{i}{r} \rfloor, \frac{j}{r} - \lfloor \frac{j}{r} \rfloor, \frac{1}{r})$$

(in order to allow the network to distinguish between different scales $\frac{1}{r}$ is inserted in the vector because for example the pixel (i, j) in a 2x image has the same projection and the same weights of the pixels $(2i, 2j)$ in a 4x image)

- **feature mapping** which use the features extracted by the *Feature Module* and the weights predicted by the *Weight Prediction* module for reconstructing the HR image:

$$\Phi(F^{LR}(i', j'), W(i, j)) = F^{LR}(i', j') * W(i, j) \quad (\text{in practice it does the convolution of the features map with the kernel for getting an intensity or rgb values})$$

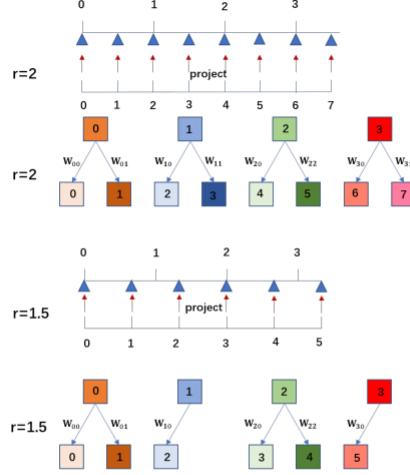


Figure 2. The schematic diagram for how to upscale the feature map with the non-integer scale factor $r = 1.5$. Here we only show the one-dimensional case for simplify.

Figure 16: Location projector: each pixel in the SR image (bottom row) has an unique pixel in the LR image (top row).

2.5.3 Results

Results with decimal scales From the results [Figure 17] Meta-SR, and in particular the **Meta upscaling module**, performs better since Meta-RDN performs better than Meta-Bi (both are learning the weights for upscaling) which perform better than BiConv (where weights are the same for each scale).

The baseline are:

- bicubic
- RDN(x1) and EDSR(x1): LR image is upscaled by r and processed by the network
- RDN(xk) and EDSR(xk): LR image is processed by the network which apply a upscaling by k then the HR image is downnscaled by $\frac{r}{k}$
- BiConv: interpolation of the final feature maps and same convolution for all scales.
- Meta-Bi: interpolation of the final feature maps but use different weights for each scale using the Weight Prediction Network.

Comparison between RDN and Meta-RDN Overall Meta-SR performs better than RDN on decimal scales.

Methods \ Scale	X1.1	X1.2	X1.3	X1.4	X1.5	X1.6	X1.7	X1.8	X1.9	X2.0
bicubic	36.56	35.01	33.84	32.93	32.14	31.49	30.90	30.38	29.97	29.55
RDN(x1)	42.41	39.76	38.00	36.68	35.57	34.64	33.87	33.19	32.60	32.08
RDN(x2)	41.84	39.34	37.87	36.63	35.56	34.63	33.83	33.1	32.52	32.11
RDN(x4)	39.71	38.48	37.33	36.29	35.34	34.52	33.81	33.14	32.60	32.09
BiConv	41.86	39.16	37.88	29.86	35.68	34.77	33.95	33.18	32.60	31.85
Meta-Bi	42.11	39.58	38.07	36.83	35.81	34.86	34.03	33.24	32.63	32.18
Meta-RDN(our)	42.82	40.40	38.28	36.95	35.86	34.90	34.13	33.45	32.86	32.35
EDSR(x1)	42.42	39.79	38.08	36.73	35.65	34.73	33.83	33.27	32.67	32.15
EDSR(x2)	41.79	39.11	37.79	36.51	35.40	34.49	33.81	33.11	32.57	32.09
EDSR(x4)	39.61	38.41	37.27	36.24	35.30	34.46	33.75	33.09	32.56	32.04
Meta-EDSR(our)	42.72	39.92	38.16	36.84	35.78	34.83	34.06	33.36	32.78	32.26
Methods \ Scale	X2.1	X2.2	X2.3	X2.4	X2.5	X2.6	X2.7	X2.8	X2.9	X3.0
bicubic	29.18	28.87	28.57	28.31	28.13	27.89	27.66	27.51	27.31	27.19
RDN(x1)	31.63	31.23	30.86	30.51	30.23	29.95	29.68	29.45	29.21	29.03
RDN(x2)	31.61	31.24	30.82	30.44	30.23	29.71	29.65	29.43	29.20	29.05
RDN(x4)	31.61	31.23	30.88	30.52	30.31	29.99	29.75	29.53	29.26	29.14
BiConv	31.53	31.11	37.87	30.38	30.16	29.81	29.55	29.28	29.05	28.91
Meta-Bi	31.59	31.21	30.91	30.54	30.34	30.01	29.76	29.54	29.28	29.22
Meta-RDN(our)	31.82	31.41	31.06	30.62	30.45	30.13	29.82	29.67	29.40	29.30
EDSR(x1)	31.69	31.29	30.91	30.56	30.28	29.98	29.73	29.49	29.25	29.07
EDSR(x2)	31.57	31.15	30.81	30.47	30.22	29.91	29.66	29.45	29.19	29.09
EDSR(x4)	31.56	31.17	30.82	30.46	30.24	29.93	29.68	29.47	29.20	29.08
Meta-EDSR(our)	31.73	31.31	30.87	30.60	30.40	30.09	29.83	29.61	29.34	29.22
Methods \ Scale	X3.1	X3.2	X3.3	X3.4	X3.5	X3.6	X3.7	X3.8	X3.9	X4.0
bicubic	26.98	26.89	26.59	26.60	26.42	26.35	26.15	26.07	26.01	25.96
RDN(x1)	28.81	28.67	28.47	28.30	28.15	28.00	27.86	27.72	27.59	27.47
RDN(x2)	28.71	28.69	28.51	28.49	28.18	28.17	28.09	27.84	27.61	27.51
RDN(x4)	28.89	28.75	28.57	28.42	28.19	28.16	27.93	27.81	27.70	27.64
BiConv	28.64	28.51	28.28	28.13	27.91	27.84	27.61	27.49	27.37	27.29
Meta-Bi	28.89	28.75	28.54	28.39	28.17	28.11	27.87	27.75	27.64	27.58
Meta-RDN(our)	28.87	28.79	28.68	28.54	28.32	28.27	28.04	27.92	27.82	27.75
EDSR(x1)	28.85	28.69	28.51	28.36	28.18	28.04	27.91	27.77	27.64	27.52
EDSR(x2)	28.78	28.64	28.45	28.34	28.11	28.06	27.83	27.73	27.61	27.49
EDSR(x4)	28.82	28.69	28.49	28.35	28.13	28.09	27.85	27.73	27.63	27.56
Meta-EDSR(our)	28.95	28.82	28.63	28.48	28.27	28.21	27.98	27.86	27.75	27.67

Figure 17: Results on Meta-SR



Figure 18: Meta-SR applied for different scales to a single image.

Methods	Metric	Set14			B100			Manga109			DIV2K		
		X2	X3	X4	X2	X3	X4	X2	X3	X4	X2	X3	X4
bicubic	PSNR	30.24	27.55	26.00	29.56	27.21	25.96	30.80	26.95	224.89	31.35	28.49	26.92
	SSIM	0.8688	0.7742	0.7227	0.8431	0.7385	0.6675	0.9339	0.8556	0.7866	0.9076	0.8339	0.7774
RDN	PSNR	34.01	30.57	28.81	32.34	29.26	27.72	39.18	34.13	31.00	35.17	31.39	29.34
	SSIM	0.9212	0.8468	0.7871	0.9017	0.8093	0.7419	0.9780	0.9484	0.9151	0.9483	0.8931	0.8446
Meta-RDN	PSNR	34.04	30.55	28.84	32.35	29.30	27.75	39.18	34.14	31.03	35.18	31.42	29.36
	SSIM	0.9213	0.8466	0.7872	0.9019	0.8096	0.7423	0.9782	0.9483	0.9154	0.9484	0.8935	0.8448

Figure 19: Meta-SR versus RDN.

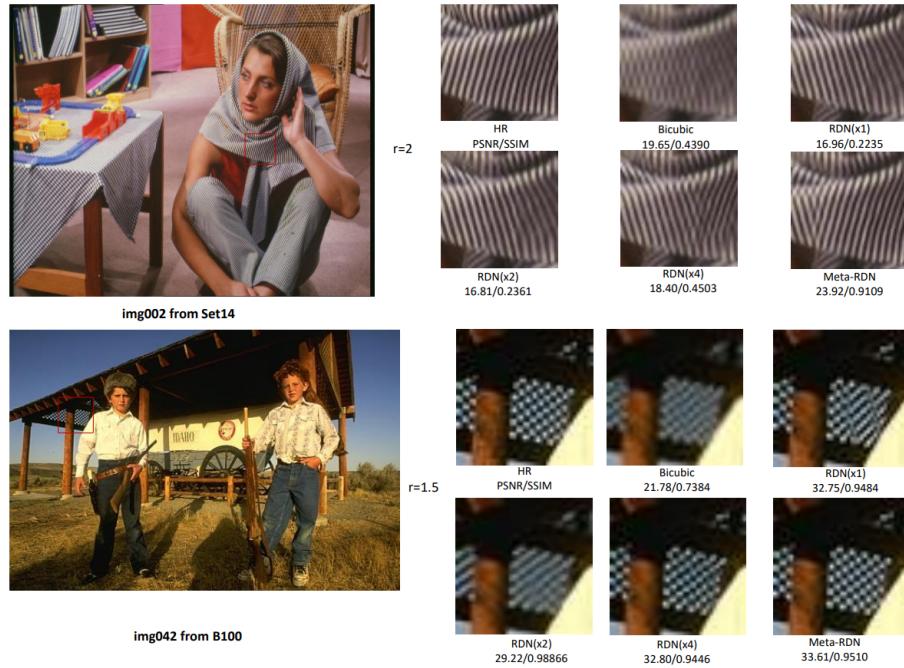


Figure 20: The visual comparison with four baselines. Our MetaRDN has better performance.

References

- [1] J. Shen, Y. Wang, and J. Zhang, “Asdn: A deep convolutional network for arbitrary scale image super-resolution,” *ArXiv*, vol. abs/2010.02414, 2020.
- [2] Y. Wang, J. Shen, and J. Zhang, “Deep bi-dense networks for image super-resolution,” *CoRR*, vol. abs/1810.04873, 2018.
- [3] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” *CoRR*, vol. abs/1807.02758, 2018.
- [4] Y. Hu, J. Li, Y. Huang, and X. Gao, “Channel-wise and spatial feature modulation network for single image super-resolution,” *CoRR*, vol. abs/1809.11130, 2018.
- [5] W. Lai, J. Huang, N. Ahuja, and M. Yang, “Deep laplacian pyramid networks for fast and accurate super-resolution,” *CoRR*, vol. abs/1704.03915, 2017.
- [6] W. Lai, J. Huang, N. Ahuja, and M. Yang, “Fast and accurate image super-resolution with deep laplacian pyramid networks,” *CoRR*, vol. abs/1710.01992, 2017.
- [7] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, “Metasr: A magnification-arbitrary network for super-resolution,” *CoRR*, vol. abs/1903.00875, 2019.
- [8] Hong Chang, Dit-Yan Yeung, and Yimin Xiong, “Super-resolution through neighbor embedding,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, pp. I–I, 2004.
- [9] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *CoRR*, vol. abs/1501.00092, 2015.
- [10] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image restoration,” *CoRR*, vol. abs/1812.10477, 2018.
- [11] J. Kim, J. K. Lee, and K. M. Lee, “Deeply-recursive convolutional network for image super-resolution,” *CoRR*, vol. abs/1511.04491, 2015.
- [12] Y. Tai, J. Yang, and X. Liu, “Image super-resolution via deep recursive residual network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2790–2798, 2017.
- [13] 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,” *CoRR*, vol. abs/1609.05158, 2016.

- [14] P. Burt and E. Adelson, “The laplacian pyramid as a compact image code,” *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, 1983.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” *CoRR*, vol. abs/1603.05027, 2016.
- [16] Q. Fan, D. Chen, L. Yuan, G. Hua, N. Yu, and B. Chen, “Decouple learning for parameterized image operators,” *CoRR*, vol. abs/1807.08186, 2018.