Alma Mater Studiorum University of Bologna

Artificial Intelligence

Machine Learning for Computer Vision

Project work

Alessandro Dicosola [Matr. 935563]

Task

SR is an Ill-posed problem: trying to create a mapping between LR to HR for reconstructing SR images
Keywords: super-sampling, super-resolution.

Related works

SR through internal and external database[2, Releated works]

Exploit similarity between textures in images therefore create pairs of LR-HR patches for learning a low dimensional space were apply k-NN, non linear regressor, random forest, ... ,
manifold embedding[1, 2004]:

- LR patches embedding is the concatenation of the first and second order derivative of the luminance at each pixel on the patch in the YIQ domain.
- HR patches embedding is the concatenation of all luminance at each pixel
- Given a LR patch find the k-NN patches on the training set

$$N_q$$

- Compute the reconstruction weights minimizing the reconstruction error:

$$argmin_W ||x_t^q - \sum_{x_s^p \in N_q} w_{qp} x_s^p||^2$$

- Reconstruct the HR patch:

$$y_t^q = \sum_{x_s^p \in N_q} w_{qp} y_s^p$$

Figure 6. 3X magnification of the bird image: (a) input low-resolution image; (b) true high-resolution image; (c) median filtering; (d) cubic spline interpolation; (e) our method with training images shown in Figures 2(c) and (d).

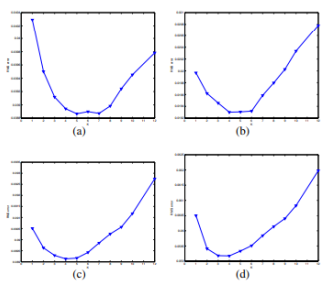


Figure 8. RMS error between the super-resolution image generated and the ground-truth image as a function of the number of nearest neighbors used: (a) head image; (b) lizard image; (c) bird image; (d) plant image.

Related works

CNN [10, Releated works] [11]

- SRCNN [3] : CNN using only three convolutions for extracting information, process them and reconstruct the SR image.
- VDSR : VGG-net that learns a residual instead of the direct mapping LR-HR in order to focus the training on high frequency information.
- SRDenseNet, RDN, DBDN[4]: Combine residual connection (locally and globally) with dense connection
- RCAN[5], CSFM[6]: Use channel-wise and spatial attention for focusing the training on important features or region in order to ease the convergence and improve the results.
- LapSRN and MS-LapSRN[2][7] : multi scale network using a *Laplacian Pyramid Framework* Laplacian Pyramid[8]
- MetaSR[9] : use a meta-upscaling module for learning LR-HR mapping using any scale.

ASDN - Arbitrary-Scale Deep Network

- Find an LR-HR mapping with any scale (integer or real)
- Use Laplacian Frequency Representation for reconstructing SR image interpolating two nearest level in the LFR.
- The range scales was found to be optimal between 1 and 2 using 11 levels in the LFR.
- For scale greater than 2 the network is *deployed recursively*.

ASDN - Arbitrary-Scale Deep Network

Laplacian Frequency Representation

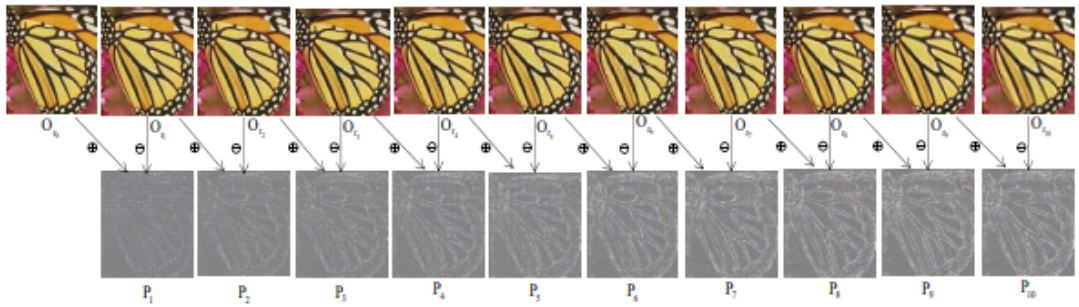


Fig. 2: The Laplacian Frequency Representation has 11 Laplacian pyramid levels ($O_{r_0}, ..., O_{r_{10}}$), with 10 phases in the scale range of $(1,2]$ ($P_1, ..., P_{10}$). Each phase represents the difference between the successive Laplacian pyramid levels.

- Learning any decimal scale is not possible: infinite decimal scales.
- Allow to have reduced dataset.
- Reconstruct the SR image interpolating the two nearest level in the LFR.
- Each level learn the HR representation at particular scale in the range.
- Compute the *phase* as the difference between two consecutive levels in order to acquire **high frequency information**:

$$P_i = O_{r_{i-1}} - O_{r_i}$$

ASDN - Arbitrary-Scale Deep Network

Laplacian Frequency Representation

- The scale for each level is:

$$r_l = \frac{l}{L-1} + 1$$

therefore the level can be indexed as

$$l = (r_l - 1) * (L - 1)$$

$$i = \lceil (r - 1) * (L - 1) \rceil$$

- Define the weight, proportional to the distance between the scales, as:

$$w_r = (L - 1) * (r_i - r)$$

- The SR image is reconstructed as:

$$O_r = O_{r_i} + w_r * P_i$$

ASDN - Arbitrary-Scale Deep Network

Example

$$r = 1.27$$

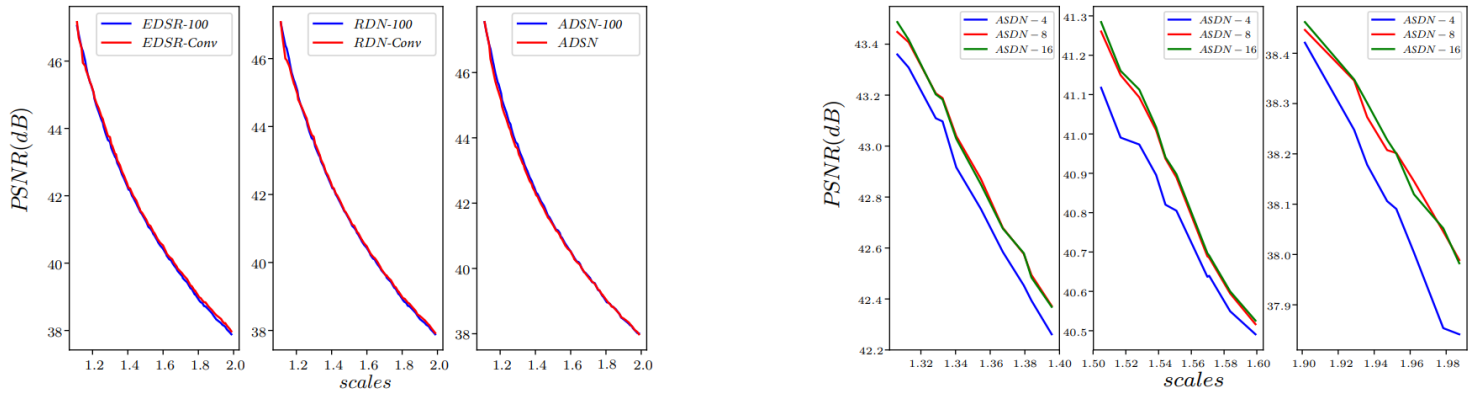
$$i = \lceil 10 * (1.27 - 1) \rceil = \lceil 2.7 \rceil = 3$$

$$w_r = 10 * (1.3 - 1.27) = 0.3$$

$$O_r = O_{r_3} + 0.3 * P_3$$

ASDN - Arbitrary-Scale Deep Network

Study of Laplacian Frequency Representation



(a) Study of Effectiveness

(b) Study of Density

Fig. 8: Study of Laplacian Frequency Representation

- ESDR-100, RDN-100, ASDN-100: upsampling networks trained and tested on 100 scales between (1,2] on Set5.
- ESDR-Conv, RDN-Conv, ADSN: networks trained as ASDN with 11 IRBs.
- ASDN-4,8,16: networks with 5,9,17 levels in the LFR with a scales range between (1,2]
- A denser LFR is better but saturate beyond 10 phases (11 levels).

ASDN - Arbitrary-Scale Deep Network

Recursive deployment

- Learning the mapping for any scale is too complex.
- For scales greater than the maximum in the scale range apply recursively the network.
- A large scale R can be expressed as an integer N power of decimal ratio r:

$$R = r^N \Rightarrow N = \log_r R$$

- Was found that using the greatest r at the beginning lead to better performance
- Since the range is fixed between (1,2] then

$$\lceil N = \log_2 R \rceil$$

and therefore

$$r_i = \begin{cases} 2 & \text{i } \leq \text{N - 1} \\ \frac{R}{2^{N-1}} & \text{i} = \text{N} \end{cases}$$

ASDN - Arbitrary-Scale Deep Network

Study of recursive deployment - Effectivness

Methods	Direct deployment			Recursive deployment		
	×2	×3	×4	×2	×3	×4
VDSR-Conv	37.57	33.77	31.56	36.86	33.70	31.50
EDSR-Conv	38.04	34.45	32.29	37.18	34.32	32.26
RDN-Conv	38.05	34.46	32.31	37.27	34.38	32.23
Ours	38.12	34.52	32.28	37.35	34.43	32.27

Table 3: PSNR of the recursive deployment and direct deployment on SR for $\times 2, \times 3, \times 4$

- *recursive deployment*: VDSR-Conv, EDSR-Conv, RDN-Conv, ASDN trained on 11 decimal scales in (1,2] and tested with HR images upscaled twice

$$\sqrt{2}, \sqrt{3}, \sqrt{4}$$

  in order to generate X2, X3, X4 HR images (therefore 2 recursions).
- *direct deployment*: VDSR-Conv, EDSR-Conv, RDN-Conv, ASDN trained on images upsampled directly X2,X3,X4.
- **Recursive deployment** lead to worst performance but the difference goes down with the scale.

ASDN - Arbitrary-Scale Deep Network

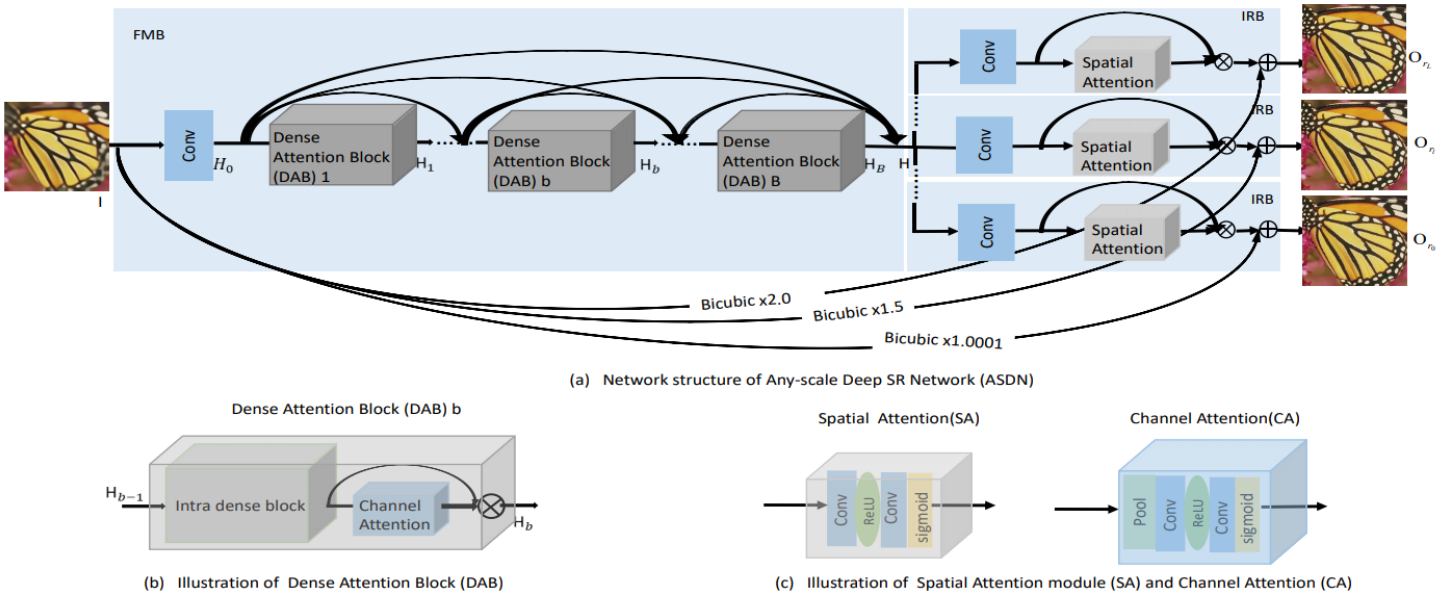
Study of recursive deployment - Optimal N and r

Scale(R)	Recursion(N)	UpscaleRatio(r)	PSNR
3×	2	1.732, 1.732	34.43
		1.500, 2.000	34.19
		2.000, 1.500	**34.48**
	3	1.442, 1.442, 1.442	33.18
4×	2	2.000, 2.000	**32.27**
	3	1.587, 1.587, 1.587	31.96
		1.800, 1.800, 1.234	32.16
		2.000, 1.800, 1.100	32.24

- Different combination of N and r.
- Use the greatest r in the scales range at the beginning lead to better performance.

ASDN - Arbitrary-Scale Deep Network

Architecture



(a) Network structure of Any-scale Deep SR Network (ASDN)

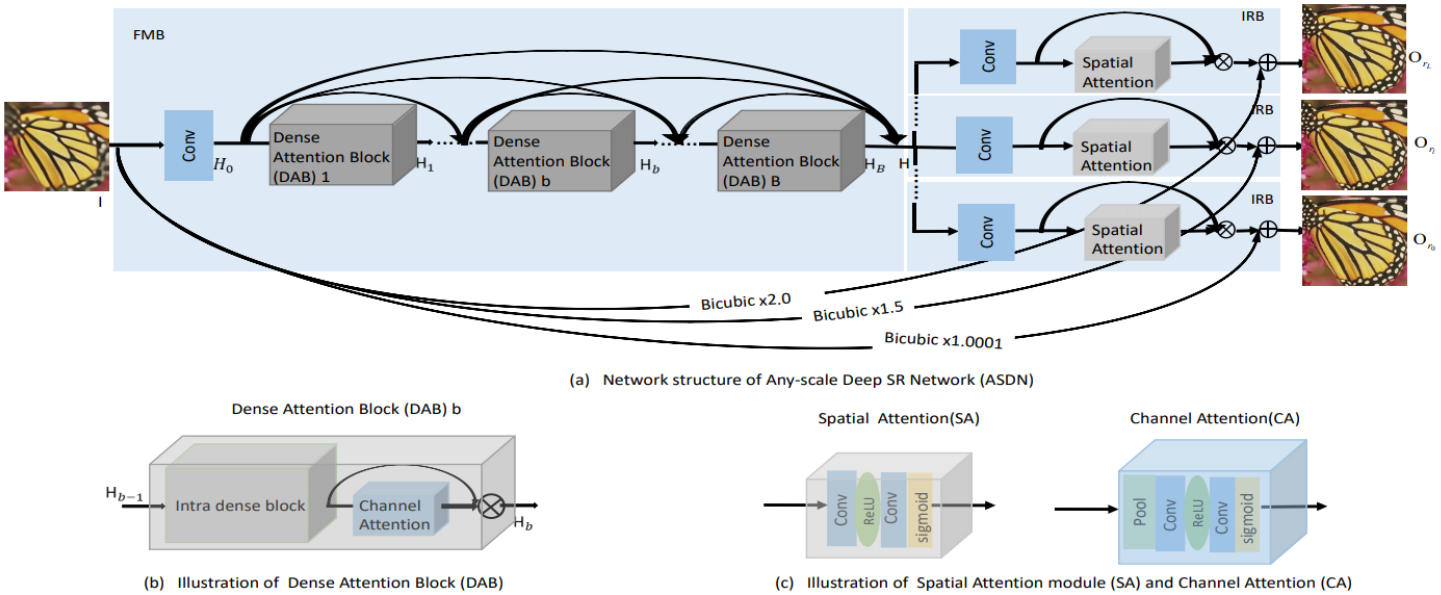
(b) Illustration of Dense Attention Block (DAB)

(c) Illustration of Spatial Attention module (SA) and Channel Attention (CA)

- Bi-dense connections (DAB with IDB) for learning efficiently features (since DenseNet, SRDenseNet and RDN doesn't allow previous features to be used into next blocks).
- IDB contains convolutions whose activations are processed with ReLu (without BN in order to maintain image details), densely connected convolutions and convolutions for compressing the input and the output (which is summed with the compressed input).
- CA from RCAN in order to focus on most important features.
- SA from CSFM in order to focus on particular and important textures at the reconstruction.
- Two IRBs are activated based on the scale used for training or testing and the the final image is reconstructed using the formula seen before.
- Global skip connection for including low frequency information from the input and allow the network to learn high frequency information only.

ASDN - Arbitrary-Scale Deep Network

Architecture



(a) Network structure of Any-scale Deep SR Network (ASDN)

(b) Illustration of Dense Attention Block (DAB)

(c) Illustration of Spatial Attention module (SA) and Channel Attention (CA)

- All convolutions have 64 filters and 3x3 kernel size with 0-padding and stride equals to 1 but the ones in IRB (1x1 kernel and 3 filters), CA (1x1 kernel which reduce-expand the channels), SA (1x1 kernel which expand-reduce the channels).
- Number of DABs: 16.
- Number of convolutions inside IDBs: 8.

ASDN - Arbitrary-Scale Deep Network

Training

- Dataset used: DIV2K and Flicker
- At each update each IRB is randomly selected and combined after FMB
- The input LR images are bicubic interpolated from the training images with 11 decimal ratios r in evenly distributed in the range of (1,2].
- In each training batch, 16 augmented RGB patches with the size 48x48 are extracted from LR images as the input, and the LR images are randomly selected from one scale training samples among the total 11 scales training data.
- In the training batch, a batch of 96x96 size patches is used as targets and the corresponding scale LR RGB patches to optimize the specific scale modules.
- Data augmentation: horizontal flip and 90-degree rotation.
- Use Adam (default parameters) with L1 loss and lr is halved at every 200000 minibatch updates for 1000000 total minibatch updates.

ASDN - Arbitrary-Scale Deep Network

Testing

- Dataset used: Set5, Set14, BSD100, Urban100 and Manga109.
- Testing images are first downscaled with the test scale s (randomly selected).
- If the scale s is not large than 2, the LR images with scale s are upsampled and forwarded into the ASDN with two enabled neighboring Laplacian pyramid levels of the scale s and HR image are predicted interpolating this two levels.
- If the scale s is large than 2, the testing is done recursively

$$\lceil \log_2 R \rceil$$

FSDN - Fixed-Scale Deep Network

- Fine-tune ASDN for a specific scale.
- Same structure of ASDN.
- A deconvolutional layer is inserted before IRB.
- Training: LR images downscaled by 2x,3x,4x,8x based on the specific scale and trained on the specific scale.
- Testing: LR images downscaled and forwarded into the network for returning the corresponding output.

ASDN : Results

Fixed-scale

Table 1: Quantitative evaluation of state-of-the-art SR algorithms. We report the average PSNR/SSIM for 2×, 3×, 4× and 8× SR. Red indicates the best performance, and blue indicates the best performance among predefined upsampling methods.

scale	Algorithms	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
2×	Bicubic	33.64	0.929	30.22	0.868	29.55	0.842	26.66	0.841	30.84	0.935
	SRCNN [4]	36.65	0.954	32.29	0.903	31.36	0.888	29.52	0.895	35.72	0.968
	VDSR [12]	37.53	0.958	32.97	0.913	31.90	0.896	30.77	0.914	37.16	0.974
	DRRN [7]	37.74	0.959	33.23	0.914	32.05	0.897	31.23	0.919	37.52	0.976
	LapSRN [13]	37.52	0.959	33.08	0.913	31.80	0.895	30.41	0.910	37.27	0.974
	MemNet [21]	37.78	0.959	33.28	0.914	32.08	0.898	31.33	0.919	37.72	0.974
	SRMDNF [27]	37.79	0.960	33.32	0.916	32.05	0.899	31.33	0.920	38.07	0.976
	ASDN(ours)	38.12	0.961	33.82	0.919	32.30	0.901	32.47	0.931	39.16	0.978
	EDSR [15]	38.11	0.960	33.92	0.920	32.32	0.901	32.93	0.935	39.10	0.976
	RDN [29]	38.24	0.961	34.01	0.921	32.34	0.902	32.96	0.936	39.19	0.978
	DBPN [6]	38.09	0.961	33.85	0.920	32.27	0.900	32.55	0.932	38.89	0.978
	RCAN [28]	38.27	0.961	34.12	0.922	32.41	0.903	33.34	0.938	39.44	0.979
	FSDN(ours)	38.27	0.961	34.18	0.923	32.41	0.903	33.13	0.937	39.49	0.979
3×	Bicubic	30.39	0.867	27.53	0.774	27.20	0.738	24.47	0.737	26.99	0.859
	SRCNN [4]	32.75	0.909	29.30	0.822	28.41	0.786	26.25	0.801	30.59	0.914
	VDSR [12]	33.66	0.921	29.77	0.831	28.82	0.798	27.41	0.830	32.01	0.934
	DRRN [7]	34.03	0.924	29.96	0.835	28.95	0.800	27.53	0.764	32.42	0.939
	LapSRN [13]	33.82	0.922	29.87	0.832	28.82	0.798	27.07	0.828	32.21	0.935
	MemNet [21]	34.09	0.925	30.00	0.835	28.96	0.800	27.57	0.839	32.51	0.937
	SRMDNF [27]	34.12	0.925	30.04	0.838	28.97	0.802	27.57	0.839	33.00	0.940
	ASDN(ours)	34.48	0.928	30.35	0.843	29.18	0.808	28.45	0.858	33.87	0.947
	EDSR [15]	34.65	0.928	30.52	0.846	29.25	0.809	28.80	0.865	34.17	0.948
	RDN [29]	34.71	0.929	30.57	0.847	29.26	0.809	28.80	0.865	34.13	0.948
	RCAN [28]	34.74	0.930	30.65	0.848	29.32	0.811	29.09	0.870	34.44	0.949
	FSDN(ours)	34.75	0.930	30.63	0.848	29.33	0.811	28.98	0.868	34.53	0.950
4×	Bicubic	28.42	0.810	26.10	0.704	25.96	0.669	23.15	0.660	24.92	0.789
	SRCNN [4]	30.49	0.862	27.61	0.754	26.91	0.712	24.53	0.724	27.66	0.858
	VDSR [12]	31.35	0.882	28.03	0.770	27.32	0.730	25.18	0.750	28.82	0.886
	DRRN [7]	31.68	0.889	28.21	0.772	27.38	0.728	25.44	0.764	29.18	0.891
	MemNet [21]	31.74	0.889	28.26	0.772	27.40	0.728	25.50	0.763	29.42	0.894
	SRMDNF [27]	31.96	0.892	28.35	0.778	27.49	0.734	25.68	0.773	30.09	0.902
	ASDN(ours)	32.27	0.896	28.66	0.784	27.65	0.740	26.27	0.792	30.91	0.913
	LapSRN [13]	31.54	0.885	28.19	0.772	27.32	0.727	25.21	0.756	29.46	0.890
	EDSR [15]	32.46	0.896	28.80	0.788	27.71	0.742	26.64	0.803	31.02	0.915
	RDN [29]	32.47	0.899	28.81	0.787	27.72	0.742	26.61	0.803	31.00	0.915
	DBPN [6]	32.42	0.898	28.76	0.786	27.68	0.740	26.38	0.796	30.91	0.914
	RCAN [28]	32.63	0.900	28.87	0.789	27.77	0.744	26.82	0.809	31.22	0.917
	FSDN(ours)	32.63	0.900	28.89	0.789	27.79	0.744	26.79	0.807	31.44	0.919
8×	Bicubic	24.40	0.658	23.10	0.566	23.97	0.548	20.74	0.516	21.47	0.650
	SRCNN [4]	25.33	0.690	23.76	0.591	24.13	0.566	21.29	0.544	22.46	0.695
	VDSR [12]	25.93	0.724	24.26	0.614	24.49	0.583	21.70	0.571	23.16	0.725
	LapSRN [13]	26.15	0.738	24.35	0.620	24.54	0.586	21.81	0.581	23.39	0.735
	MemNet [21]	26.16	0.741	24.38	0.619	24.58	0.584	21.89	0.583	23.56	0.738
	ASDN(ours)	27.02	0.776	24.99	0.641	24.82	0.600	22.57	0.620	24.73	0.748
	EDSR [15]	26.96	0.776	24.91	0.642	24.81	0.599	22.51	0.622	24.69	0.784
	DBPN [6]	27.21	0.784	25.13	0.648	24.88	0.601	22.73	0.631	25.14	0.799
	RCAN [28]	27.31	0.788	25.23	0.651	24.98	0.606	23.00	0.645	25.24	0.803
	FSDN(ours)	27.33	0.789	25.24	0.651	24.98	0.604	22.90	0.638	25.24	0.803

- Overall FSDN is better than SOTA networks up to now.
- RCAN is sometimes better due to more channel attention layers which allow better reconstruction.
- ASDN is better than other networks in all fixed scales although it is trained in the range (1,2].
- Sometime ASDN performs worst than other networks because they are trained on specific scales with specific upsampled modules (transposed convolution) not usable with decimal ratios. The meta-upsampling module can be used only on scales used for training. Indeed FSDN performs better than ASDN due to the transposed convolution.

ASDN : Results

Any-scale

Table 2: Results of any-scale SR on different methods tested on BSD100. The first row shows the results of Laplacian pyramid levels and the second row demonstrates SR performance on randomly selected scales and boundary condition. Red indicates the best performance, and blue indicates the second best performance.

Method \ Scale	X1.1	X1.2	X1.3	X1.4	X1.5	X1.6	X1.7	X1.8	X1.9
Bicubic	36.56	35.01	33.84	32.93	32.14	31.49	30.90	30.38	29.97
VDSR-Conv	42.13	39.52	37.88	36.53	35.42	34.50	33.72	33.03	32.41
EDSR-Conv	42.92	40.11	38.33	36.93	35.79	34.85	34.06	33.38	32.75
RDN-Conv	42.86	40.04	38.25	36.86	35.72	34.78	33.99	33.29	32.67
Meta-EDSR [8]	42.72	39.92	38.16	36.84	35.78	34.83	34.06	33.36	32.78
Meta-RDN [8]	42.82	40.40	38.28	36.95	35.86	34.90	34.13	33.45	32.86
ASDN(ours)	43.05	40.24	38.42	37.02	35.87	34.92	34.14	33.46	32.86

Method \ Scale	X2.0	X2.8	X4.0	X5.7	X8.0	X11.3	X16.0	X22.6	X32.0
Bicubic	29.55	27.53	25.96	24.96	23.67	22.65	21.73	20.73	19.90
VDSR-Conv	31.89	29.23	27.25	25.56	24.58	23.49	22.47	21.39	20.38
EDSR-Conv	32.23	29.54	27.58	26.01	24.78	23.65	22.63	21.55	20.53
RDN-Conv	32.07	29.47	27.51	25.94	24.72	23.60	22.58	21.50	20.51
Meta-EDSR [8]	32.26	29.61	27.67	-	-	-	-	-	-
Meta-RDN [8]	32.35	29.67	27.75	-	-	-	-	-	-
ASDN(ours)	32.30	29.63	27.65	26.07	24.85	23.70	22.66	21.59	20.55

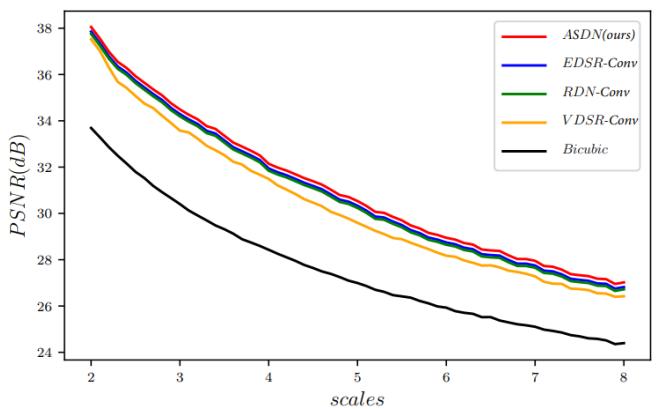


Fig. 4: PSNR comparison of ASDN with other works within the continuous scale range $(\times 2, \times 8]$ on Set5

- VDSR-Conv, EDSR-Conv,RDN-Conv are networks trained as ADSN
- Meta-EDSR, Meta-RDN are networks with a meta-upscaling module trained with scales range between [1,4] with stride 0.1
- ASDN and networks trained as ASDN perform better on continuos (random) scales between 1 and 8.

ASDN : Results

Qualitative comparison

Fig. 5: Qualitative comparisons of our models with other works on $\times 2$ super-resolution. Red indicates the best performance, and blue indicates the second best

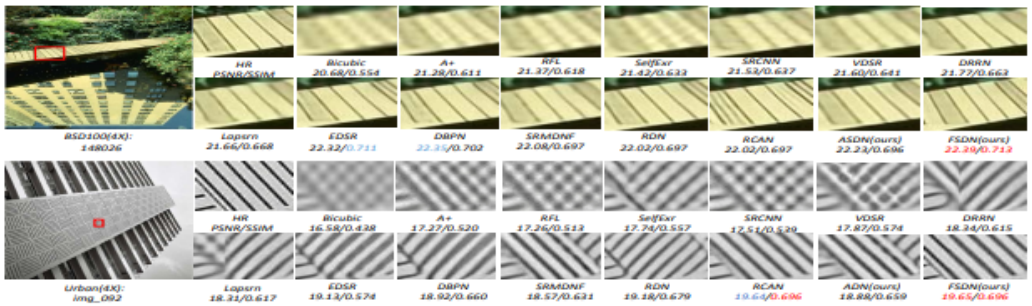


Fig. 6: Qualitative comparisons of our models with other works on $\times 4$ super-resolution. Red indicates the best performance, and blue indicates the second best

Fig. 7: Qualitative comparisons of our models with other works on $\times 8$ super-resolution. Red indicates the best performance, and blue indicates the second best

ASDN

Model analysis

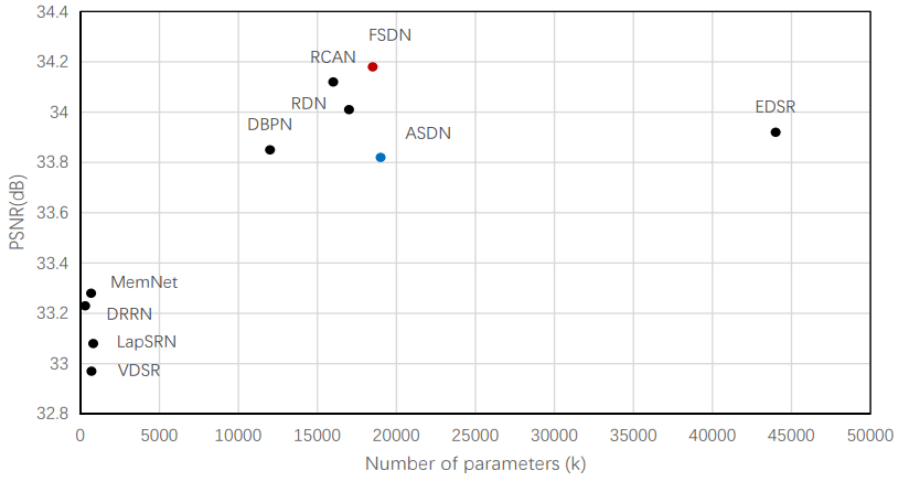


Fig. 9: Performance vs number of parameters. The results are evaluated with Set14 for 2× enlargement. Red indicates the best performance, and blue indicates the best performance among predefined upsampling methods

Module	Different combination of CA, SA and SC							
CA	×	×	×	√	√	√	×	√
SA	×	×	√	×	√	×	√	√
SC	×	√	×	×	×	√	√	√
PSNR	37.92	37.96	37.93	37.95	37.97	37.99	37.97	**38.01**

Table 5: Investigation of channel attention (CA), spatial at-

Extra

Laplacian pyramid[8]

- Generate images with high frequency information only subtracting filtered images with itself non filtered: similar to apply a laplacian operator on the image.
- Remove correlation between pixels.
- Those images can be used for reconstructing the original image.
- Hand crafted kernel for filtering the image in order to cutoff low frequency (low-pass filtering) information:
  - Separable
  - Symmetric
  - Normalized:

$$a + 2b + 2c = 1$$

  - Equal contribution:

$$a + 2c = 2b$$

  - Kernel =

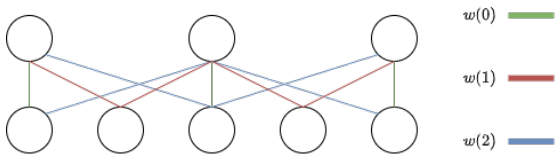


Figure: Equal contribution.

- All nodes in the lower level must have the same contribution to nodes in the upper level.
- The central node contributes with a + 2c
- The middle ones contribute with 2b
- The outer ones with a + c
- So a+2c=2b

Extra

Laplacian pyramid[8]

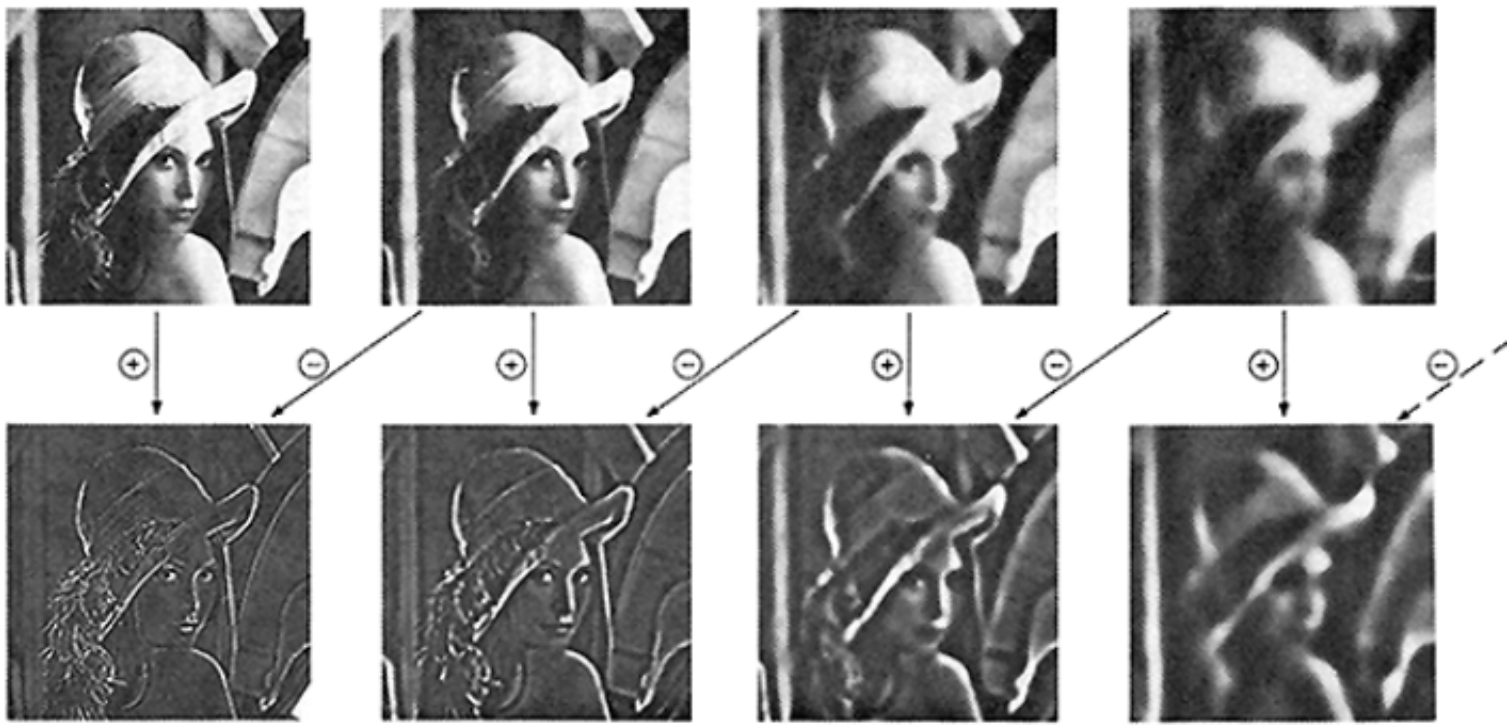


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtainedby expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

Extra

DBDN: Deep Bi-Dense Networks for image SR [4]

- Deeper network lead to better performance
- The deeper the network is the more likely the vanishing/exploding gradient is present.
- Use skip connections (with sum).
- Reuse features thanks to dense connection (with concatenation) for improving final results.
- The flow is limited from one block to another: features are not really reused.

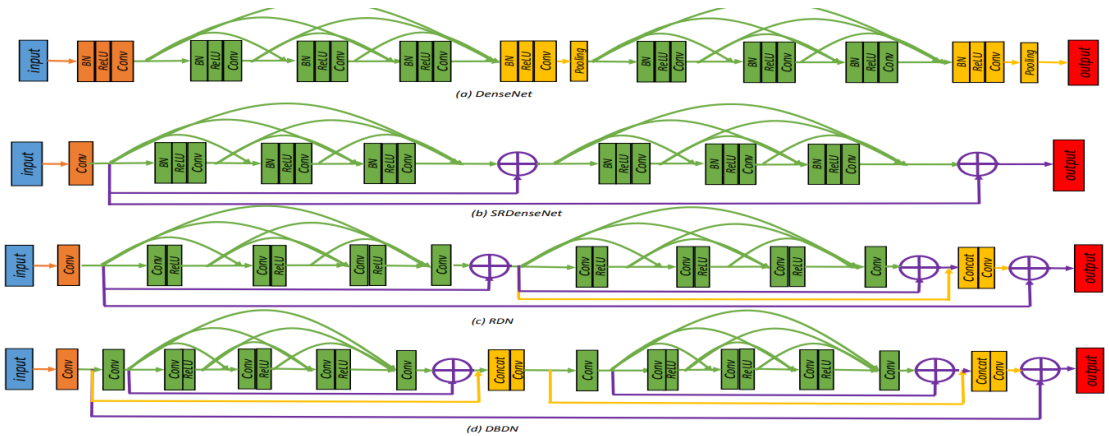


Fig. 2. Simplified structure of (a) DenseNet [26]. The green lines and layers denote the connections and layers in the dense block, and the yellow layers denote the transition and pooling layer. (b) SRDenseNet [15]. The green layers are the same dense block structures as those in DenseNet. The purple lines and element-wise $\oplus$ refer to the skip conection. (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.

Extra

DBDN: Deep Bi-Dense Networks for image SR [4]

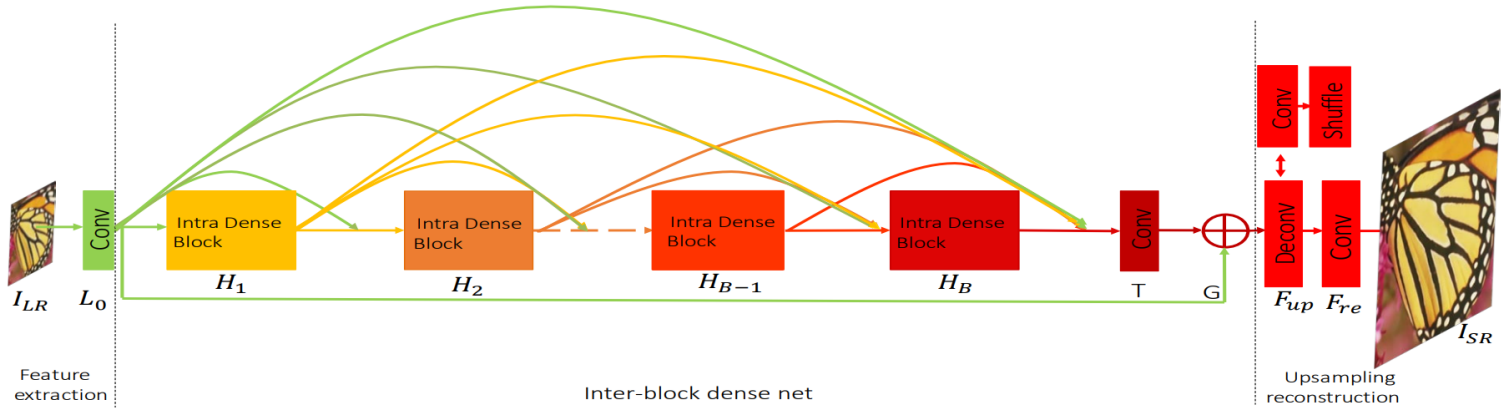


Fig. 3. Network structure

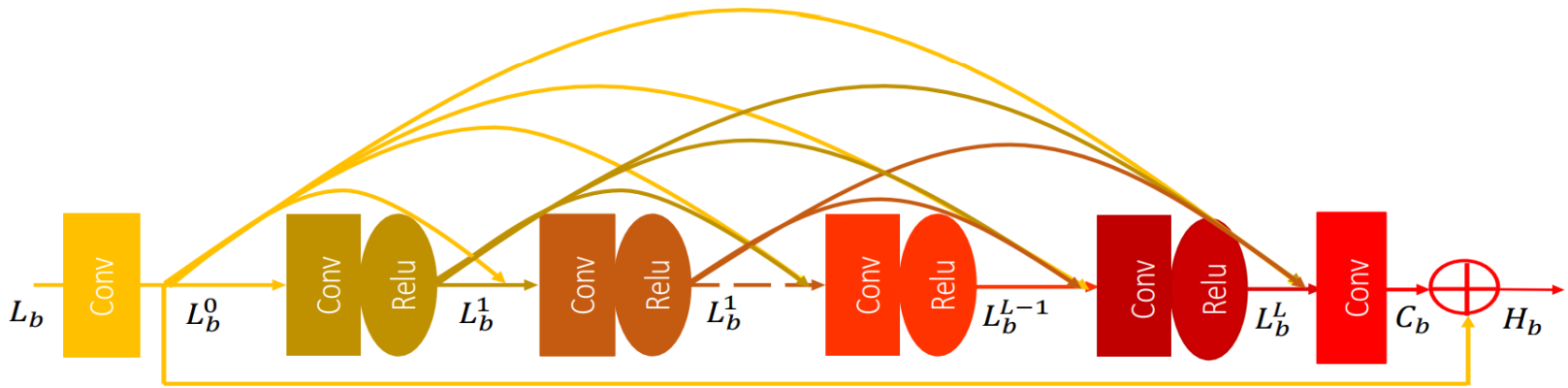


Fig. 4. Intra dense blocks

Extra

Meta-SR: A Maginfication-Arbitrary Network for SR [9]

- SR for a specific integer scale as independent task: memory-wise and time-wise costly (since you have to train and maintain data and models) .
- The same hold with decimal scale.
- Meta-SR exploit **meta-learning** creating **meta-upscale** module which allow to learn to generate weights based on the input scale and therefore use one network for all possible *trained* scales.

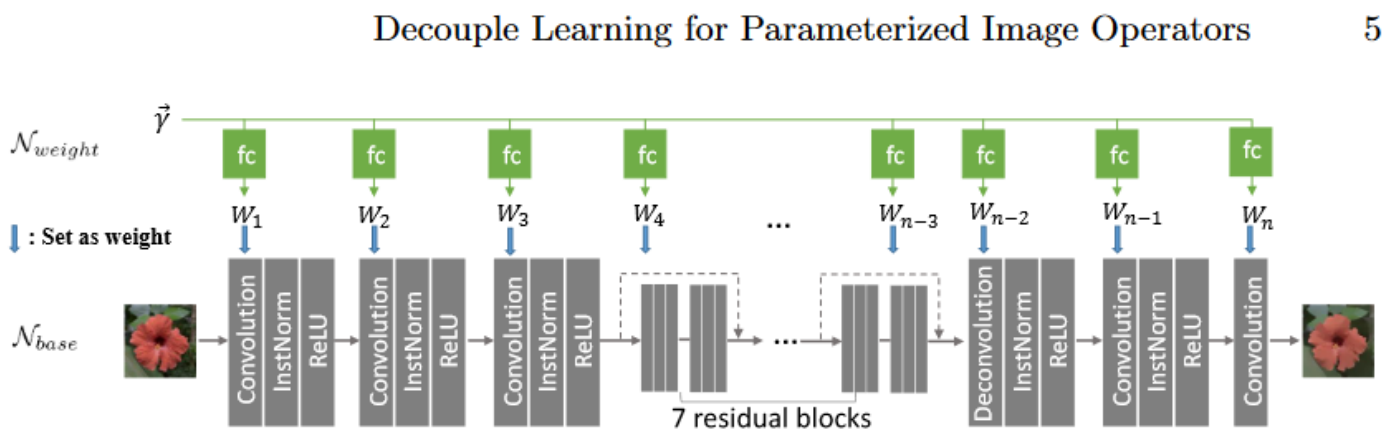
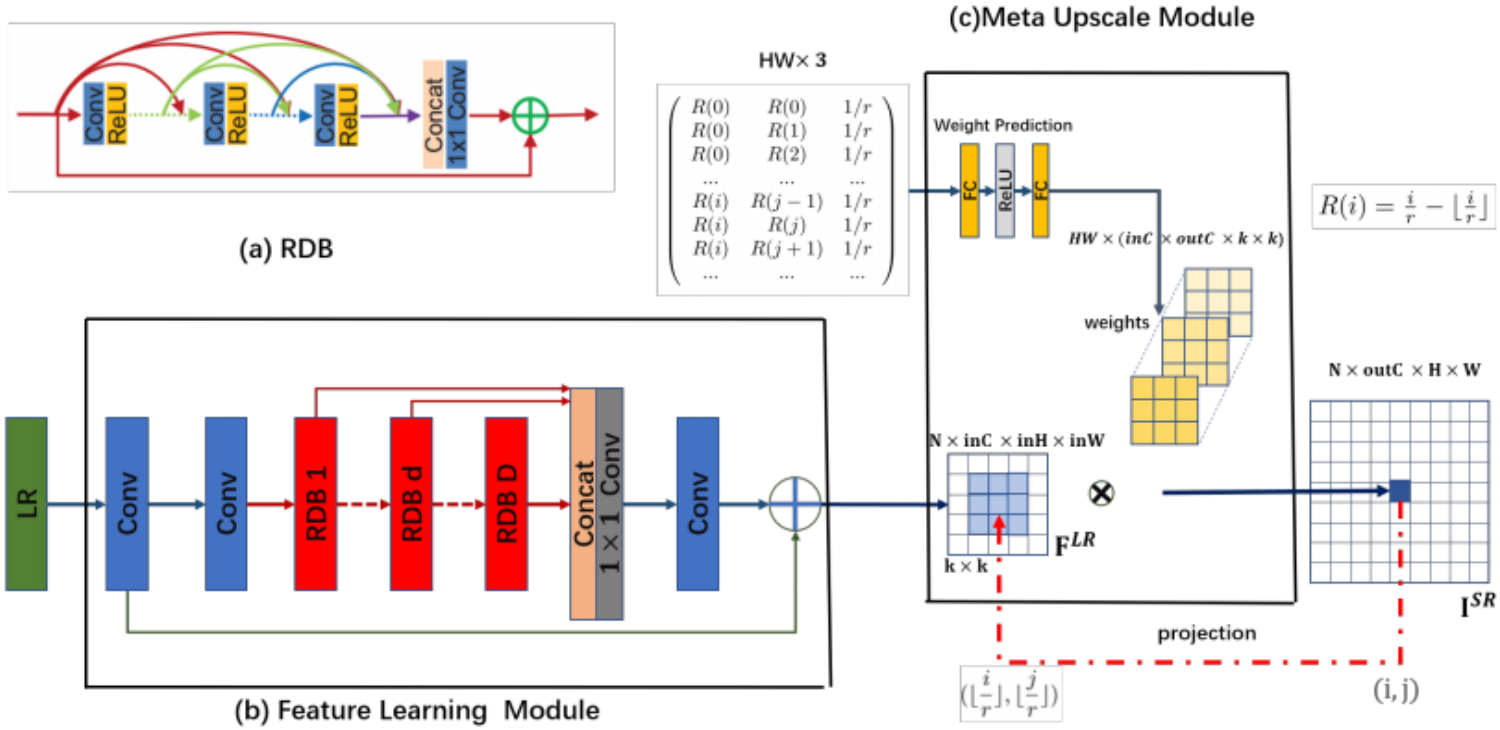


Figure: Meta-learning on Parametrized Image Operators [12]

Extra

Meta-SR: A Maginfication-Arbitrary Network for SR [9]



Meta Upscale Module:

- *location projector*: project pixel in HR to LR image
- *weight prediction*: predicts the weights of the kernels using a fully connected network (2 layers with 256 hidden units and ReLU as activation) The input is vector of offset (proper choice) and scales (in order to differentiate different scales during training therefore learn different weight for different scales).
- *feature mapping*: reconstruct the image as the matrix product between the features extracted and the weights computed.

Extra

RCAN: Image SR using very deep Residual Channel Attention Networks [5]

- Deep networks are very effective for CV tasks
- This lead to vanishing/exploding gradients, hence skip connections are used in order to avoid these.
- The LR image contains low-frequency information
- The processed features inside the network contain high-frequency information.
- Increase the performance focusing the learning of high-frequency information using **attention**.

Extra

RCAN: Image SR using very deep Residual Channel Attention Networks [5]

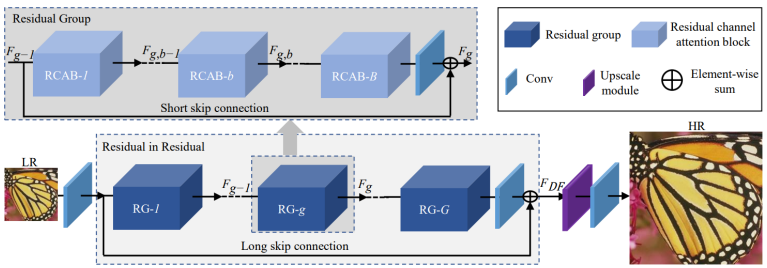


Fig. 2. Network architecture of our residual channel attention network (RCAN)

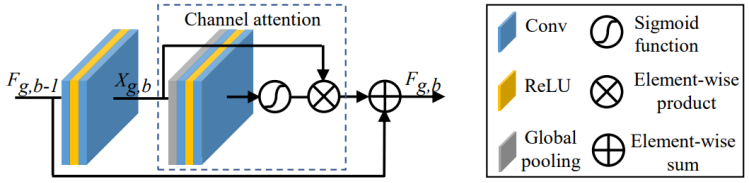


Fig. 4. Residual channel attention block (RCAB)

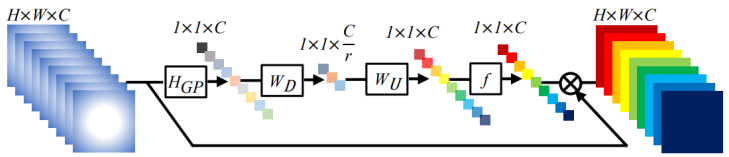


Fig. 3. Channel attention (CA). ⊗ denotes element-wise product

- **Residual-in-Residual**: allow to build a deeper network as well as allow to flow low-frequency information from shallow levels to deeper levels easing the training.
- **Channel-wise attention**: Increase the performance of the network focusing on most important channel-wise features.

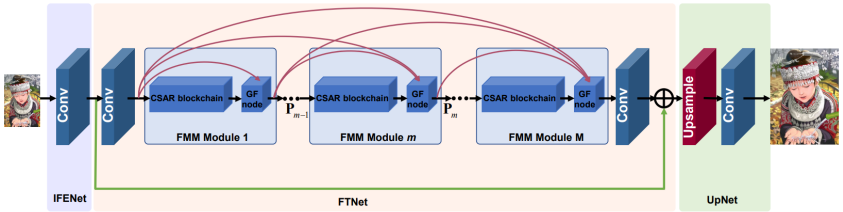
Extra

CSFM: Channel-wise and Spatial Feature Modulation Network for Single Image SR [6]

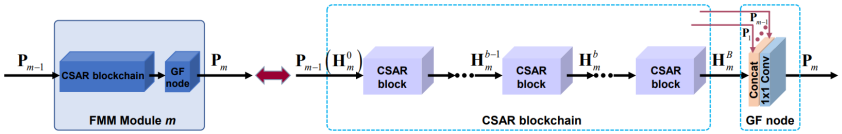
- Use deeper network for higher performance.
- Use residual and dense connections for avoiding exploding/vanishing gradient.
- Use residual and dense connection for features reusing and flowing most important information in the network.
- Exploit attention channel-wise and spatial-wise for learning high-frequency information and information on most important region.

Extra

CSFM: Channel-wise and Spatial Feature Modulation Network for Single Image SR [6]

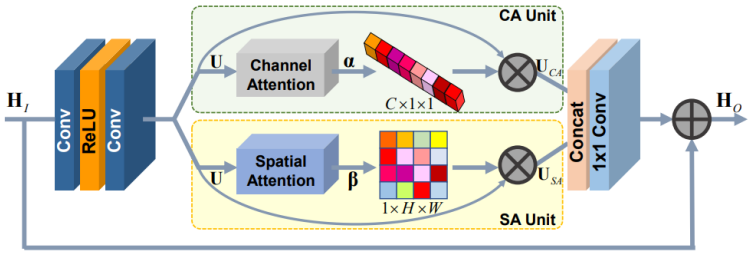


(a) The architecture of the proposed CSFM network

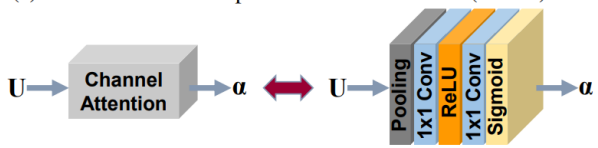


(b) The structure of feature-modulation memory (FMM) module in CSFM network

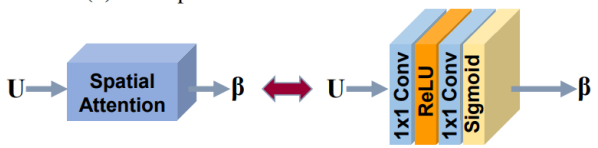
Fig. 2: The architecture of our CSFM network and the structure of FMM module in CSFM network. (a) The overall architecture of the proposed CSFM network, which adopts adaptive feature-modulation strategy, long-term information persistence mechanism and post-upscaling scheme to boost SR performance. (b) The feature-modulation memory (FMM) module in (a), which exploits a chain of channel-wise and spatial attention residual (CSAR) blocks to capture more informative features and utilizes the gated fusion (GF) node to fusion long-term information from the preceding FMM modules and short-term information from the current module.



(a) Channel-wise and spatial attention residual (CSAR) block



(b) The operations of channel-wise attention



(c) The operations of spatial attention

- Use FMMs for exploiting channel and spatial information as well as create a memory using the GF node.
- The CSAR blockchain allow to capture most important features information and flow these into the FMM module.
- The GF node concatenate the current CSAR blockchain output and all previous FMM outputs for preserving long-term information and also create new features for SR image reconstruction.

Extra

LapSR and MSLapSR: Fast and accuracte Image SR with Deep Laplacian Pyramid Networks [2] [7]

LapSRN

- L2 loss fail to capture one-to-many mapping between LR and HR patches: use **Charbonnier loss**: $L_S = \frac{1}{N}\sum_{i=1}^{N}\sum_{l=1}^{L}\rho \times \left(\left(y_l^{(i)} - x_l^{(i)}\right) - r_l^{(i)}\right)$ where S is the target upsampling factor, $L = \log_2 S$, $\rho = \sqrt{x^2 + \epsilon^2}$ (Charbonnier penalty function, differentiable variant of L1 norm).
- Apply the upsampling operator at the end (previous papers apply it at the beginning increasing inference and training time): **transposed convolution** or **subpixel convolution**.
- Single upsampling step lead to worst performance: **reconstruct progressively** the SR image using a laplacian pyramid framework [8].

MS-LapSRN

- **Parameter sharing** within (reuse of the same recursive blok) pyramid levels and across pyramid levels (reuse of the same feature extractor).
- Use of **skip connections** for avoiding vanishing/exploding gradients
- **Multi-scale training** (in the previous work they trained different network for each scale: 2x,4x,8x)

Extra

LapSR and MSLapSR: Fast and accuracte Image SR with Deep Laplacian Pyramid Networks [2] [7]

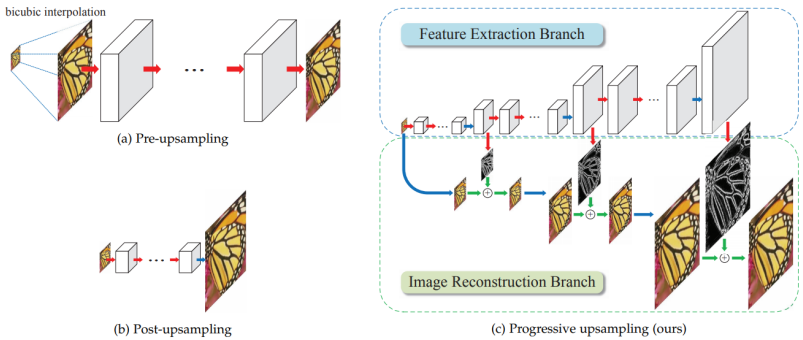


Fig. 1. **Comparisons of upsampling strategies in CNN-based SR algorithms.** Red arrows indicate convolutional layers. Blue arrows indicate transposed convolutions (upsampling), and green arrows denote element-wise addition operators. (a) Pre-upsampling based approaches (e.g., SRCNN [9], VDSR [11], DRCN [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.

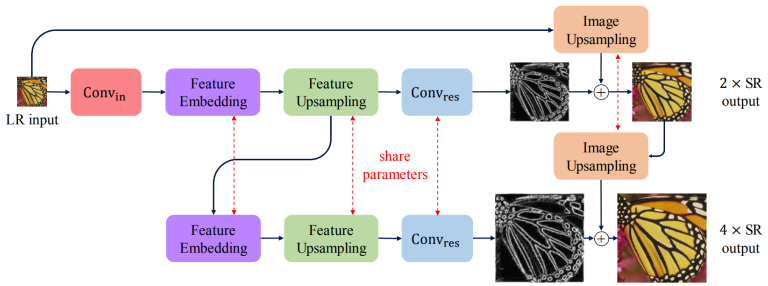


Fig. 3. **Detailed network architecture of the proposed LapSRN.** At each pyramid level, our model consists of a feature embedding sub-network for extracting non-linear features, transposed convolutional layers for upsampling feature maps and images, and a convolutional layer for predicting the sub-band 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.

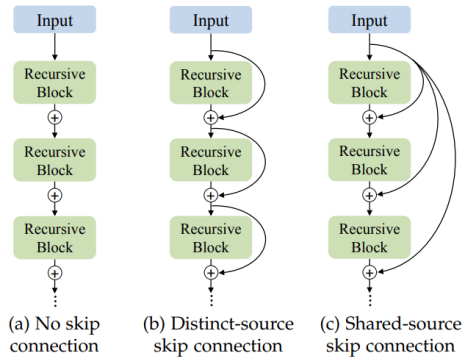


Fig. 4. **Local residual learning.** We explore three different ways of local skip connection in the feature embedding sub-network of the LapSRN for training deeper models.

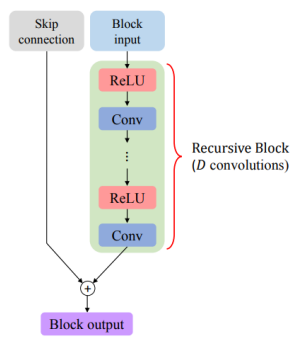


Fig. 5. **Structure of our recursive block.** There are D convolutional layers in a recursive block. The weights of convolutional layers are distinct within the block but shared among all recursive blocks. We use the pre-activation structure [41] without the batch normalization layer.


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.


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.


C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *CoRR*, vol. abs/1501.00092, 2015.


Y. Wang, J. Shen, and J. Zhang, "Deep bi-dense networks for image super-resolution," *CoRR*, vol. abs/1810.04873, 2018.


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.


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.


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.


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.


X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-sr: A magnification-arbitrary network for super-resolution," *CoRR*, vol. abs/1903.00875, 2019.


J. Shen, Y. Wang, and J. Zhang, "Asdn: A deep convolutional network for arbitrary scale image super-resolution," *ArXiv*, vol. abs/2010.02414, 2020.


S. Anwar, S. Khan, and N. Barnes, "A deep journey into super-resolution: A survey," *CoRR*, vol. abs/1904.07523, 2019.


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.