

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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Background</b>	<b>3</b>
2.1	Deep Bi-Dense Network (DBDN[5]) . . . . .	3
2.1.1	Architectures . . . . .	3
2.1.2	Results . . . . .	4
2.2	Residual Channel Attention Network (RCAN[8]) . . . . .	7
2.2.1	Architecture . . . . .	7
2.2.2	Results . . . . .	8
2.3	Channel-wise and Spatial Feature Modulation Network (CSFM[9])	13
2.3.1	Architecture . . . . .	13
2.3.2	Results . . . . .	14
2.4	Deep Laplacian Pyramid Network (LapSRN[11]) and Multi-scale Deep Laplacian Pyramid Network (MS-LapSRN [12]) . . . . .	19
2.4.1	Features . . . . .	19
2.4.2	Architecture . . . . .	20
2.4.3	Loss . . . . .	21
2.4.4	Depth . . . . .	21
2.4.5	Results . . . . .	21
2.5	Magnification-Arbitrary Network (MetaSR [10]) . . . . .	25
2.5.1	Background . . . . .	25
2.5.2	Architecture . . . . .	25
2.5.3	Results . . . . .	27
2.6	Extra . . . . .	31
2.6.1	The Laplacian Pyramid [14] . . . . .	31
2.6.2	Identity mapping as output of the residual block [15] . . . . .	34
2.6.3	Pixel Shuffle: sub-pixel convolution [13] . . . . .	36
<b>3</b>	<b>Any-Scale Deep Network (ASDN[1])</b>	<b>37</b>
3.1	Laplacian Frequency Representation . . . . .	37
3.2	Recursive deployment . . . . .	38
3.3	Architecture . . . . .	38
3.4	FSDN . . . . .	39
3.5	Results . . . . .	39
3.5.1	Efficiency of the Laplacian Frequency Representation and density study. . . . .	39
3.5.2	Efficiency of the recursive deployment and optimal N and r	39
3.5.3	Ablation studies and performance versus parameters . . . . .	40
3.5.4	Quantitative and qualitative results . . . . .	41
<b>4</b>	<b>Project</b>	<b>45</b>
4.1	Hardware . . . . .	45
4.2	Architecture configuration . . . . .	45
4.3	The initial LR . . . . .	46

## 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[2].

Due to advancement of deep learning, SR task started to be achieved using deep neural networks: from the simplest one, SRCNN[3] 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 [4][5] and recursive blocks [6][7], attention mechanism [8][9], meta-learning [10], multi-levels [11][12].

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 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[10] (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.

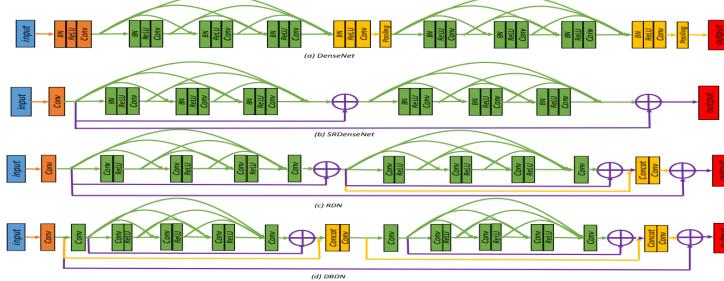


Fig. 2. Simplified structure of (a) DenseNet [26]. The green lines and layers denote the connections and layers in the dense block, and the yellow layers denote the transition and pooling layer. (b) SRDenseNet [15]. The green layers are the same dense block structures as those in DenseNet. The purple 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.

## 2 Background

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

### 2.1 Deep Bi-Dense Network (DBDN[5])

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.

#### 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):

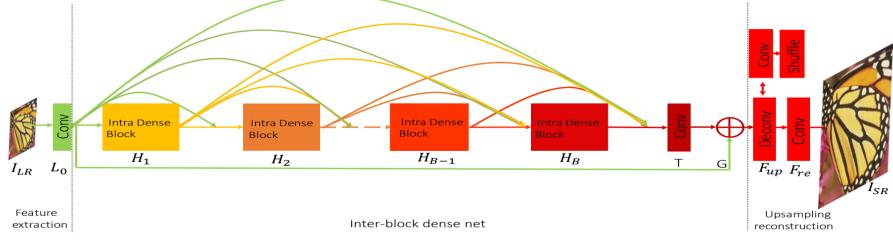


Fig. 3. Network structure

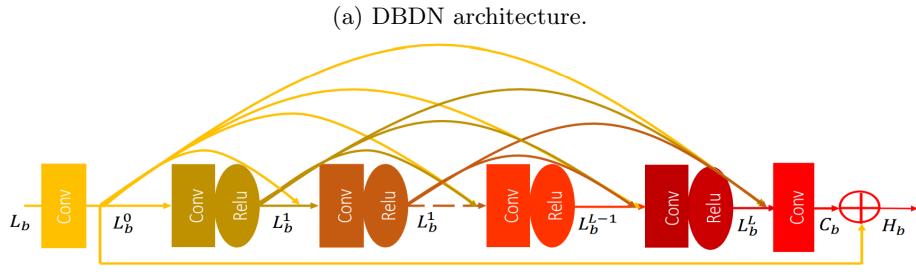


Fig. 4. Intra dense blocks

(b) Intra dense block.

- $n_r * \text{number of IDBs before.}$
- $n_r + L * n_g$  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]

### 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.

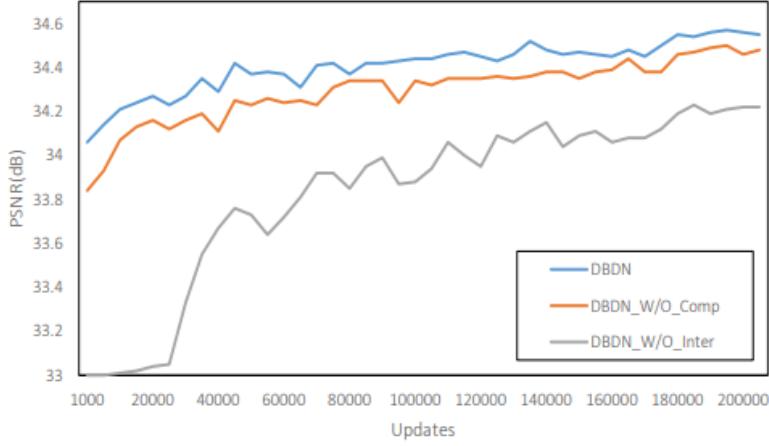


Fig. 8. Discussion about transition layer, skip connection and inter-block dense connectivity in DBDN. The results are evaluated with Set5 for  $3\times$  enlargement in 200 epochs

Figure 3: Ablation studies on DBDN.

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	37.53/0.9857	37.52/0.9591	37.74/0.9591	-/-	38.11/0.9601	38.24/0.9614	38.09/0.9600	<b>38.30/0.9617</b>	<b>38.35/0.9618</b>
	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	-/-	<b>34.76/0.9299</b>	<b>34.83/0.9303</b>
	4×	28.42/0.8104	31.35/0.8833	31.54/0.8855	31.68/0.8888	32.02/0.8934	32.46/0.8968	32.47/0.8990	32.47/0.8980	<b>32.54/0.8991</b>	<b>32.70/0.9006</b>
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	<b>34.20/0.9224</b>	<b>34.34/0.9239</b>
	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	-/-	<b>30.63/0.8478</b>	<b>30.75/0.8495</b>
	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	<b>28.89/0.7890</b>	<b>29.00/0.7908</b>
BSD100	2×	29.56/0.8431	31.90/0.8969	31.80/0.8950	32.05/0.8973	-/-	32.32/0.9013	32.34/0.9017	32.27/0.9000	<b>32.39/0.9022</b>	<b>32.45/0.9028</b>
	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	-/-	<b>29.31/0.8104</b>	<b>29.37/0.8112</b>
	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	<b>27.76/0.7426</b>	<b>27.84/0.7446</b>
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	<b>32.98/0.9364</b>	<b>33.36/0.9389</b>
	3×	24.46/0.7349	26.24/0.7988	27.14/0.8272	27.53/0.8378	-/-	28.80/0.8655	28.79/0.8655	-/-	<b>28.96/0.8682</b>	<b>29.17/0.8715</b>
	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	<b>26.70/0.8050</b>	<b>27.00/0.8117</b>
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	<b>39.46/0.9788</b>	<b>39.65/0.9793</b>
	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	-/-	<b>34.46/0.9498</b>	<b>34.80/0.9512</b>
	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	<b>31.23/0.9169</b>	<b>31.68/0.9198</b>

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

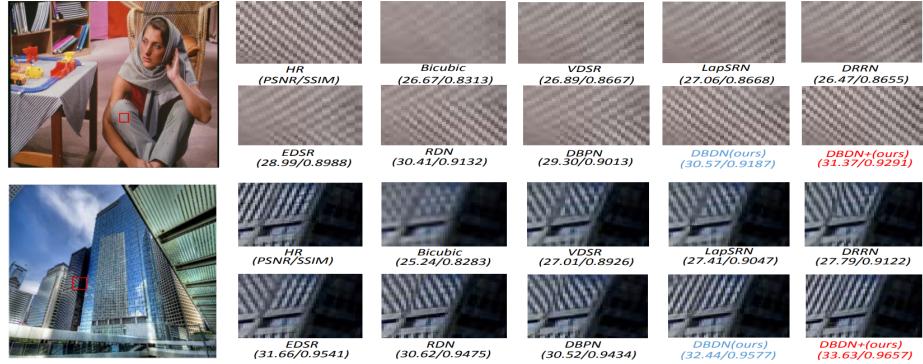


Figure 5: Qualitative results of DBDN.

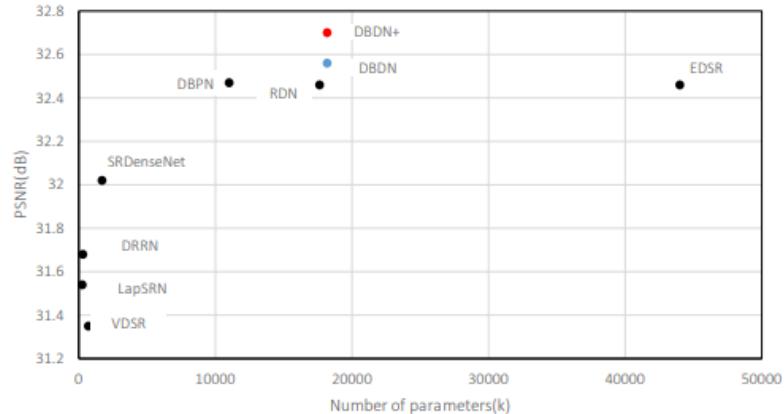
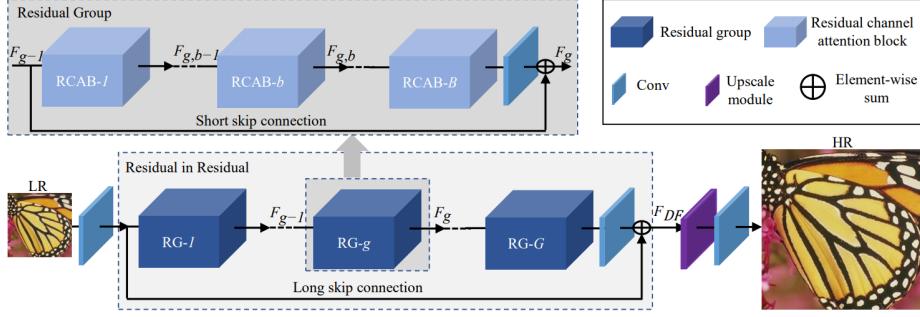


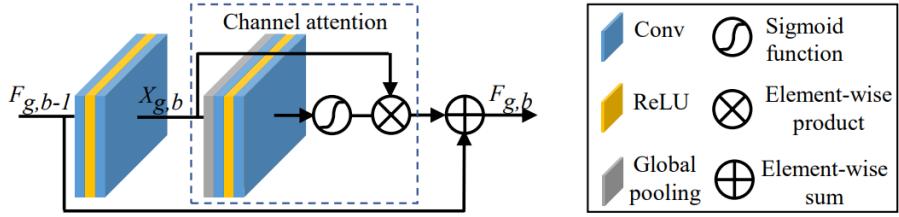
Fig. 7. Performance vs number of parameters. The results are evaluated with Set5 for 4 $\times$  enlargement. Red indicates the best performance and blue indicates the second best.

Figure 6: Performance-Parameters study.



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

Figure 7: RCAN architecture.



**Fig. 4.** Residual channel attention block (RCAB)

Figure 8: Residual channel attention block.

## 2.2 Residual Channel Attention Network (RCAN[8])

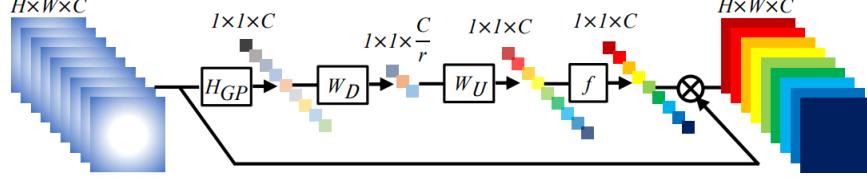
In SR task in order to reconstruct an SR image *high-frequency information* are extremely important, therefore **attention** mechanism are deployed in order to do so.

As usual the depth is also important hence the use of residuals which allow to have a deeper network avoiding exploding/vanishing gradient.

### 2.2.1 Architecture

Figure 7 highlights:

- **shallow extractor** for extracting low level features
- deep features extractor using **Residual in Residual (RIR)**: the main module is the residual groups (**RG**) created using a sequence of **RCAB**[Figure 8] where short skip connection are present which with the long skip connection allow to ease the training and reduce the amount of low-frequency information in the network because directly extracted from the LR image. There are **G** residual groups composed by **B** *residual channel attention*



**Fig. 3.** Channel attention (CA).  $\otimes$  denotes element-wise product

Figure 9: Channel attention used in RCAB.

blocks.

- **upsampling module** using *transposed convolution, nearest neighbor and convolution, subpixel convolution*.
- **reconstruction** using a single convolution.

**Channel attention** The **Channel attention** has to capture high-frequency information inside the features extracted by convolutions; in order to do so a global statistics of the features is computed using global average pooling channel-wise which are further processed in order to create a mask using a sigmoid activation which allow to create a non mutual-exclusive mask and non-linear information which allow to increase the expressing power of the network.

### 2.2.2 Results

**Ablation studies on LSC, SSC and CA** From Figure 10 it's possible to see that short-skip connections (SSC) and long-skip connections (LSC) are extremely important but in order to improve the final results channel attention is necessary.

**Table 1.** Investigations of RIR (including LSC and SSC) and CA. We observe the best PSNR (dB) values on Set5 ( $2\times$ ) in  $5\times 10^4$  iterations

Residual in Residual (RIR)	LSC	✗	✓	✗	✓	✗	✓	✗	✓
	SSC	✗	✗	✓	✓	✗	✗	✓	✓
Channel attention (CA)		✗	✗	✗	✗	✓	✓	✓	✓
PSNR on Set5 ( $2\times$ )		37.45	37.77	37.81	37.87	37.52	37.85	37.86	37.90

Figure 10: Ablation studies done on RCAN.

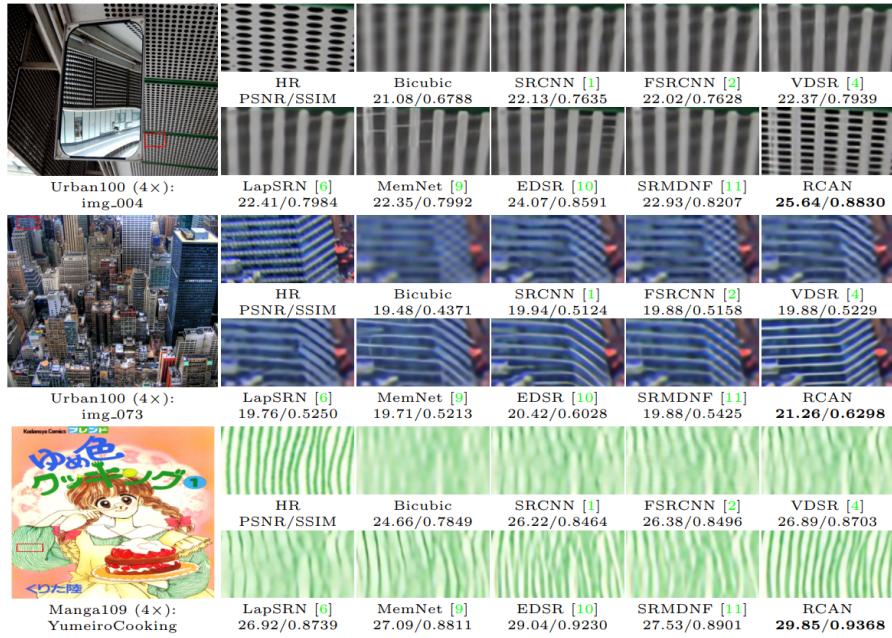
### Quantitative and qualittive results

**Using Bicubic as degradation model** Thanks to the depth and channel attention RCAN is able to reconstruct the original HR image without artifact or blurring.

**Table 2.** Quantitative results with BI degradation model. Best and second best results are highlighted and underlined

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM								
Bicubic	$\times 2$	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
SRCNN [1]	$\times 2$	<u>36.66</u>	0.9542	32.45	<u>0.9067</u>	31.36	0.8879	29.50	0.8946	35.60	0.9663
FSRCNN [2]	$\times 2$	37.05	0.9560	32.66	0.9090	31.53	0.8920	29.88	0.9020	36.67	0.9710
VDSR [4]	$\times 2$	37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140	37.22	0.9750
LapSRN [6]	$\times 2$	37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101	37.27	0.9740
MemNet [9]	$\times 2$	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	37.72	0.9740
EDSR [10]	$\times 2$	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF [11]	$\times 2$	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
D-DBPN [16]	$\times 2$	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RDN [17]	$\times 2$	38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353	39.18	0.9780
RCAN (ours)	$\times 2$	<u>38.27</u>	<u>0.9614</u>	<u>34.12</u>	<u>0.9216</u>	<u>32.41</u>	<u>0.9027</u>	<u>33.34</u>	<u>0.9384</u>	<u>39.44</u>	<u>0.9786</u>
RCAN+ (ours)	$\times 2$	<b>38.33</b>	<b>0.9617</b>	<b>34.23</b>	<b>0.9225</b>	<b>32.46</b>	<b>0.9031</b>	<b>33.54</b>	<b>0.9399</b>	<b>39.61</b>	<b>0.9788</b>
Bicubic	$\times 3$	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349	26.95	0.8556
SRCNN [1]	$\times 3$	32.75	0.9090	29.30	0.8215	28.41	0.7863	26.24	0.7989	30.48	0.9117
FSRCNN [2]	$\times 3$	33.18	0.9140	29.37	0.8240	28.53	0.7910	26.43	0.8080	31.10	0.9210
VDSR [4]	$\times 3$	33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290	32.01	0.9340
LapSRN [6]	$\times 3$	33.82	0.9227	29.87	0.8320	28.82	0.7980	27.07	0.8280	32.21	0.9350
MemNet [9]	$\times 3$	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376	32.51	0.9369
EDSR [10]	$\times 3$	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
SRMDNF [11]	$\times 3$	34.12	0.9254	30.04	0.8382	28.97	0.8025	27.57	0.8398	33.00	0.9403
RDN [17]	$\times 3$	34.71	0.9296	30.57	0.8468	29.26	0.8093	28.80	0.8653	34.13	0.9484
RCAN (ours)	$\times 3$	<u>34.74</u>	<u>0.9299</u>	<u>30.65</u>	<u>0.8482</u>	<u>29.32</u>	<u>0.8111</u>	<u>29.09</u>	<u>0.8702</u>	<u>34.44</u>	<u>0.9499</u>
RCAN+ (ours)	$\times 3$	<b>34.85</b>	<b>0.9305</b>	<b>30.76</b>	<b>0.8494</b>	<b>29.39</b>	<b>0.8122</b>	<b>29.31</b>	<b>0.8736</b>	<b>34.76</b>	<b>0.9513</b>
Bicubic	$\times 4$	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
SRCNN [1]	$\times 4$	30.48	0.8628	27.50	0.7513	26.90	0.7101	24.52	0.7221	27.58	0.8555
FSRCNN [2]	$\times 4$	30.72	0.8660	27.61	0.7550	26.98	0.7150	24.62	0.7280	27.90	0.8610
VDSR [4]	$\times 4$	31.35	0.8830	28.02	0.7680	27.29	0.7026	25.18	0.7540	28.83	0.8870
LapSRN [6]	$\times 4$	31.54	0.8850	28.19	0.7720	27.32	0.7270	25.21	0.7560	29.09	0.8900
MemNet [9]	$\times 4$	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630	29.42	0.8942
EDSR [10]	$\times 4$	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF [11]	$\times 4$	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
D-DBPN [16]	$\times 4$	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
RDN [17]	$\times 4$	32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028	31.00	0.9151
RCAN (ours)	$\times 4$	<u>32.63</u>	<u>0.9002</u>	<u>28.87</u>	<u>0.7889</u>	<u>27.77</u>	<u>0.7436</u>	<u>26.82</u>	<u>0.8087</u>	<u>31.22</u>	<u>0.9173</u>
RCAN+ (ours)	$\times 4$	<b>32.73</b>	<b>0.9013</b>	<b>28.98</b>	<b>0.7910</b>	<b>27.85</b>	<b>0.7455</b>	<b>27.10</b>	<b>0.8142</b>	<b>31.65</b>	<b>0.9208</b>
Bicubic	$\times 8$	24.40	0.6580	23.10	0.5660	23.67	0.5480	20.74	0.5160	21.47	0.6500
SRCNN [1]	$\times 8$	25.33	0.6900	23.76	0.5910	24.13	0.5660	21.29	0.5440	22.46	0.6950
FSRCNN [2]	$\times 8$	20.13	0.5520	19.75	0.4820	24.21	0.5680	21.32	0.5380	22.39	0.6730
SCN [3]	$\times 8$	25.59	0.7071	24.02	0.6028	24.30	0.5698	21.52	0.5571	22.68	0.6963
VDSR [4]	$\times 8$	25.93	0.7240	24.26	0.6140	24.49	0.5830	21.70	0.5710	23.16	0.7250
LapSRN [6]	$\times 8$	26.15	0.7380	24.35	0.6200	24.54	0.5860	21.81	0.5810	23.39	0.7350
MemNet [9]	$\times 8$	26.16	0.7414	24.38	0.6199	24.58	0.5842	21.89	0.5825	23.56	0.7387
MSLapSRN [7]	$\times 8$	26.34	0.7558	24.57	0.6273	24.65	0.5895	22.06	0.5963	23.90	0.7564
EDSR [10]	$\times 8$	26.96	0.7762	24.91	0.6420	24.81	0.5985	22.51	0.6221	24.69	0.7841
D-DBPN [16]	$\times 8$	27.21	0.7840	25.13	0.6480	24.88	0.6010	22.73	0.6312	25.14	0.7987
RCAN (ours)	$\times 8$	<u>27.31</u>	<u>0.7878</u>	<u>25.23</u>	<u>0.6511</u>	<u>24.98</u>	<u>0.6058</u>	<u>23.00</u>	<u>0.6452</u>	<u>25.24</u>	<u>0.8029</u>
RCAN+ (ours)	$\times 8$	<b>27.47</b>	<b>0.7913</b>	<b>25.40</b>	<b>0.6553</b>	<b>25.05</b>	<b>0.6077</b>	<b>23.22</b>	<b>0.6524</b>	<b>25.58</b>	<b>0.8092</b>

Figure 11: Quantitative results of RCAN.



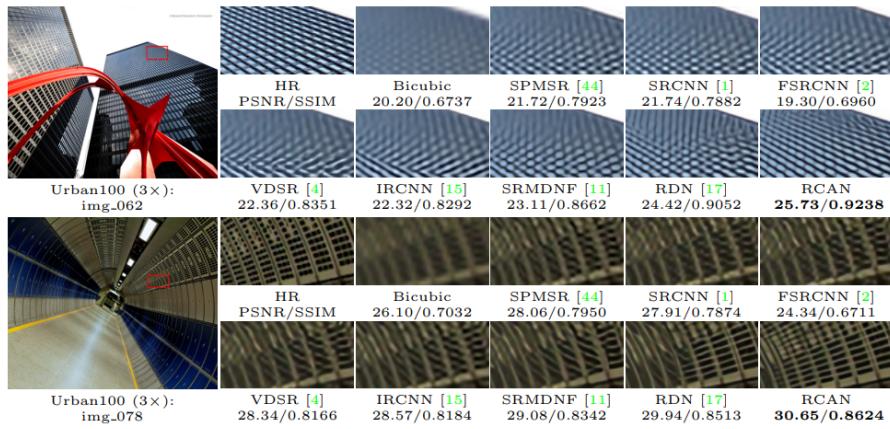
**Fig. 5.** Visual comparison for  $4 \times$  SR with BI model on Urban100 and Manga109 datasets. The best results are highlighted

Figure 12: Qualitative results of RCAN.

**Using Blur-Down degradation model** RCAN performs better than any state-of-the-art.

**Table 3.** Quantitative results with BD degradation model. Best and second best results are **highlighted** and underlined

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM								
Bicubic	×3	28.78	0.8308	26.38	0.7271	26.33	0.6918	23.52	0.6862	25.46	0.8149
SPMSR [44]	×3	32.21	0.9001	28.89	0.8105	28.13	0.7740	25.84	0.7856	29.64	0.9003
SRCNN [1]	×3	32.05	0.8944	28.80	0.8074	28.13	0.7736	25.70	0.7770	29.47	0.8924
FSRCNN [2]	×3	26.23	0.8124	24.44	0.7106	24.86	0.6832	22.04	0.6745	23.04	0.7927
VDSR [4]	×3	33.25	0.9150	29.46	0.8244	28.57	0.7893	26.61	0.8136	31.06	0.9234
IRCNN [15]	×3	33.38	0.9182	29.63	0.8281	28.65	0.7922	26.77	0.8154	31.15	0.9245
SRMDNF [11]	×3	34.01	0.9242	30.11	0.8364	28.98	0.8009	27.50	0.8370	32.97	0.9391
RDN [17]	×3	34.58	0.9280	30.53	0.8447	29.23	0.8079	28.46	0.8582	33.97	0.9465
RCAN (ours)	×3	34.70	0.9288	30.63	0.8462	29.32	0.8093	28.81	0.8647	34.38	0.9483
RCAN+ (ours)	×3	<b>34.83</b>	<b>0.9296</b>	<b>30.76</b>	<b>0.8479</b>	<b>29.39</b>	<b>0.8106</b>	<b>29.04</b>	<b>0.8682</b>	<b>34.76</b>	<b>0.9502</b>



**Fig. 7.** Visual comparison for 3× SR with BD model on Urban100 dataset. The best results are **highlighted**

Figure 13: Quantitative and qualitative results of RCAN using blur as degradation.

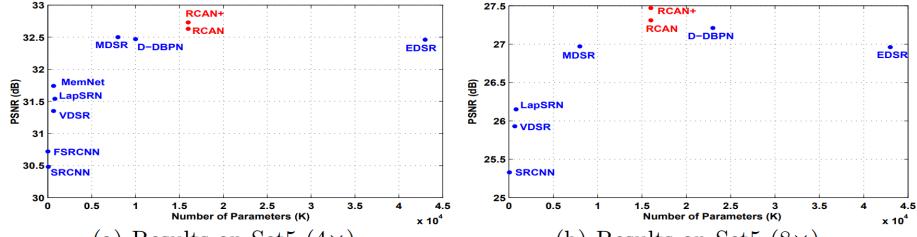
**With object recognition task** Using RCAN for upscaling an 56x56 image to 256x256 achieves the lowest top-1 and top-5 errors in object detection.

**Table 4.** ResNet object recognition performance. The best results are **highlighted**

Evaluation	Bicubic	DRCN [19]	FSRCNN [2]	PSyCo [45]	ENet-E [8]	RCAN	Baseline
Top-1 error	0.506	0.477	0.437	0.454	0.449	<b>0.393</b>	0.260
Top-5 error	0.266	0.242	0.196	0.224	0.214	<b>0.167</b>	0.072

Figure 14: RCAN used at the beginning of a pipeline (upsamples an image to 224x244) for object recognition

**Performance versus Parameters** RCAN achieve the best performance with not so many parameters.



**Fig. 8.** Performance and number of parameters. Results are evaluated on Set5

Figure 15: Comparison of parameters and performance of state-of-the-art SR network.

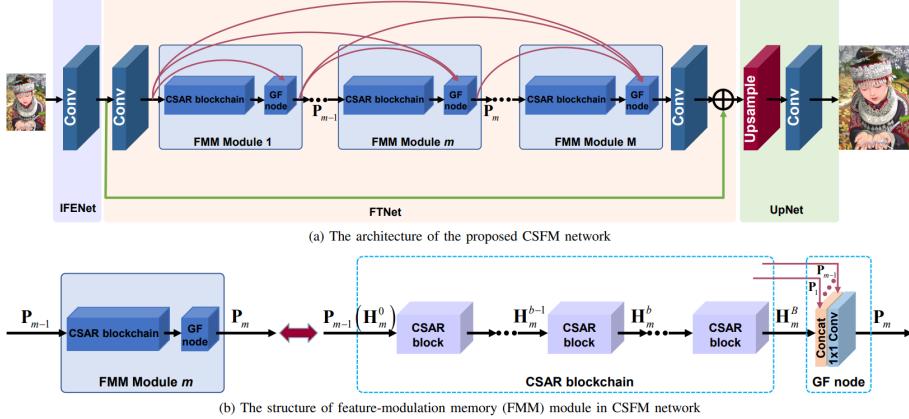


Fig. 2: The architecture of our CSFM network and the structure of FMM module in CSFM network. (a) The overall architecture of the proposed CSFM network, which adopts adaptive feature-modulation strategy, long-term information persistence mechanism and post-upscaling scheme to boost SR performance. (b) 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.

Figure 16: CSFM architecture.

## 2.3 Channel-wise and Spatial Feature Modulation Network (CSFM[9])

CSFM exploit features reuse with dense connections, ease the training with skip and dense connections as well as avoid the flow of low-level features inside the network and focus on high-frequency information and on particular spatial position using channel-wise and spatial-wise attention mechanisms.

### 2.3.1 Architecture

**Feature-Modulation Memory** CSFM contains stacked **feature-modulation memory (FMM)** which contains a **CSAR blockchain** and a **GF node**.

Since the **gated-function node (GF node)** processes not only the features received by the **CSAR blockchain** but also the ones from the output of all previous FMMs, it is preserving long-term features information.

These information flow from one FMM to another but also in the network from shallow levels to deeper ones thanks to the dense connection, this ease the training but also allow to create features at multiple levels.

The **CSAR blockchain** is a sequence of **Channel and spatial attention residual block (CSAR)**[Figure 17].

**Channel and spatial attention residual block** CSAR is a residual block where there is an identity skip connection that flow low-frequency information, ease the training and preserve shotr-term information and a residual path where

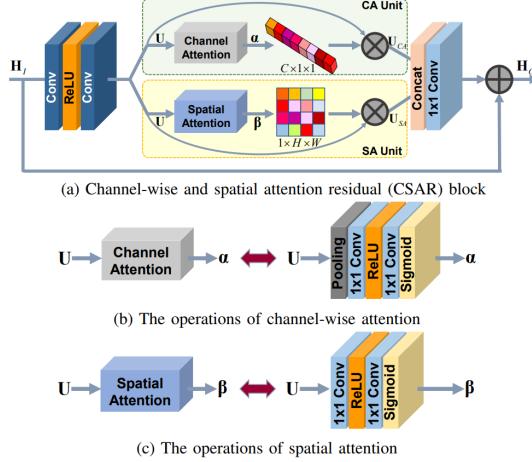


Figure 17: CSAR block.

a **channel attention** and **spatial attention** mechanisms are used for extracting high-frequency information feature-wise and spatial-wise.

**CA and SA** The **channel attention**(*CA*) and **spatial attention**(*SA*) are both used for creating a mask in order to improve the expressive power of the network.

The *CA* extract a channel-wise statistics from the residual processed in *CSAR* using global averaging pooling and processing non-linear features with a sigmoid.

The *SA* extract a spatial-wise statistics simply expanding the channel with the convolution, extracting non-linear features, reducing the channel to the original number and then computing the mask with the same spatial dimension as the input.

### 2.3.2 Results

**Ablation studies** The [Figure 18] have proved the efficacy of the CA, SA (since they are able to increase high-frequency information) and GF (since it's able to process long-term information along the network).

Components		Different Combinations of Components						
In residual blocks	Channel-wise attention (CAR)	✗	✓	✗	–	✗	✓	✗
	Spatial attention (SAR)	✗	✗	✓	–	✗	✗	✓
	Combinational attention of CA and SA (CSAR)	✗	–	–	✓	✗	–	✓
Gated fusion (GF) node with long-term dense connections		✗	✗	✗	✗	✓	✓	✓
PSNR (dB)		32.38	32.48	32.44	32.54	32.48	32.52	32.50



(a) Results using BR (simple residual block), CSAR(CA and SA), CSAR and GF

Figure 18: Ablation studies for CSFM.

**Analysis of the GF node** The [Figure 19] highlights that long-term information are more important than short-term information (the ones coming from the CSAR blockchain) therefore adding GF is important.

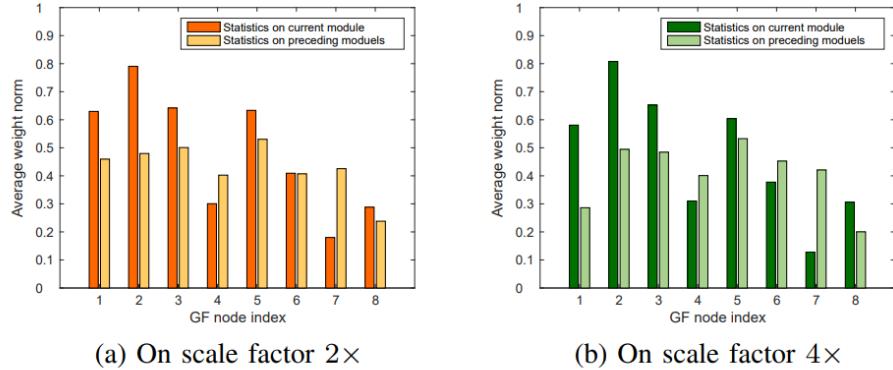


Fig. 6: The average weight norms of short-term features from the current FMM module and of long-term features from the preceding FMM modules. (a) The statistics are conducted for a scale factor of  $2\times$ . (b) The statistics are conducted for a scale factor of  $4\times$ .

Figure 19: Normalized weight norms in eight GF nodes of eight FMM modules for two scale factors of  $2\times$  and  $4\times$ .

### Number of FMM and CSAR blocks and total amount of parameters

Increasing B and M allow to have a deeper network hence better performance with low cost in number of parameters.

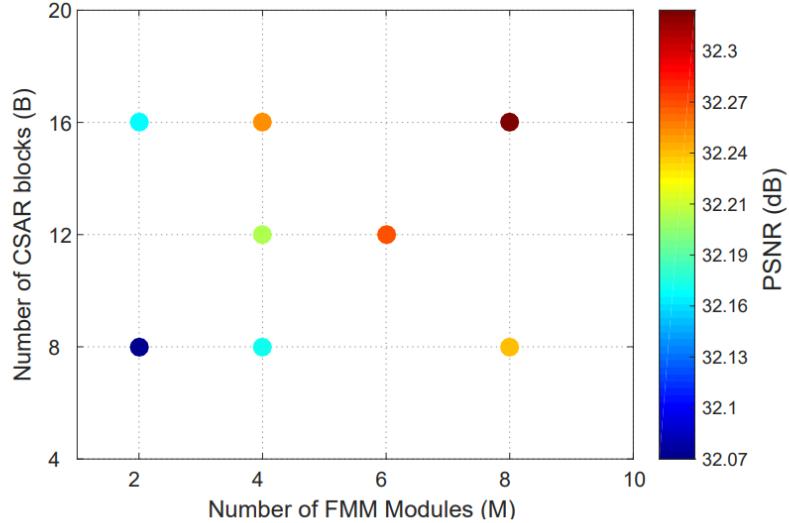


Fig. 7: PSNR performance versus the number of FMM modules (M) and the number of CSAR blocks (B) per FMM . The color of the point denotes the PSNR value that corresponds to the color bar on the right. The tests are conducted for a scale factor of  $2\times$  on the dataset of BSD100.

Figure 20: Study done on the number of FMM and CSAR blocks.

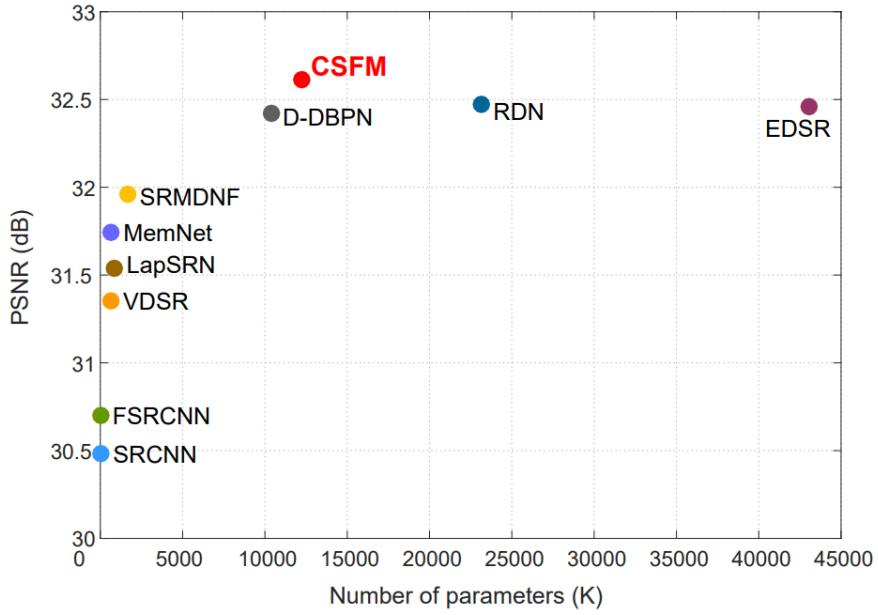


Figure 21: Number of parameters of CSFM with FMM=8 and CSAR=16

**Quantitative and qualitative results** CSFM outperforms all state-of-art-networks.

Scale	Method	SET5		SET14		BSD100		URBAN100		MANGA109	
		PSNR	SSIM								
2×	Bicubic	33.68	0.9304	30.24	0.8691	29.56	0.8435	26.88	0.8405	30.81	0.9348
	SRCNN [14]	36.66	0.9542	32.45	0.9067	31.36	0.8879	29.51	0.8946	35.70	0.9677
	FSRCNN [20]	36.98	0.9556	32.62	0.9087	31.50	0.8904	29.85	0.9009	36.56	0.9703
	VDSR [15]	37.53	0.9587	33.05	0.9127	31.90	0.8960	30.77	0.9141	37.41	0.9747
	LapSRN [27]	37.52	0.9591	32.99	0.9124	31.80	0.8949	30.41	0.9101	37.27	0.9740
	DRRN [16]	37.74	0.9591	33.23	0.9136	32.05	0.8973	31.23	0.9188	37.88	0.9750
	MemNet [19]	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	38.02	0.9755
	IDN [31]	37.83	0.9600	33.30	0.9148	32.08	0.8985	31.27	0.9196	38.02	0.9749
	EDSR [12]	38.11	0.9601	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.19	0.9782
	SRMDNF [52]	37.79	0.9601	33.32	0.9154	32.05	0.8984	31.33	0.9204	38.07	0.9761
3×	D-DBPN [29]	38.13	0.9609	33.83	0.9201	32.28	0.9009	32.54	0.9324	38.89	0.9775
	RDN [13]	38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353	39.18	0.9780
	CSFM (ours)	<b>38.26</b>	<b>0.9615</b>	<b>34.07</b>	<b>0.9213</b>	<b>32.37</b>	<b>0.9021</b>	<b>33.12</b>	<b>0.9366</b>	<b>39.40</b>	<b>0.9785</b>
	Bicubic	30.40	0.8686	27.54	0.7741	27.21	0.7389	24.46	0.7349	26.95	0.8565
	SRCNN [14]	32.75	0.9090	29.29	0.8215	28.41	0.7863	26.24	0.7991	30.56	0.9125
	FSRCNN [20]	33.16	0.9140	29.42	0.8242	28.52	0.7893	26.41	0.8064	31.12	0.9196
	VDSR [15]	33.66	0.9213	29.78	0.8318	28.83	0.7976	27.14	0.8279	32.13	0.9348
	LapSRN [27]	33.82	0.9227	29.79	0.8320	28.82	0.7973	27.07	0.8271	32.21	0.9344
	DRRN [16]	34.03	0.9244	29.96	0.8349	28.95	0.8004	27.53	0.8378	32.74	0.9388
	MemNet [19]	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376	32.79	0.9391
	IDN [31]	34.11	0.9253	29.99	0.8354	28.95	0.8013	27.42	0.8359	32.69	0.9378
	EDSR [12]	34.65	0.9282	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.20	0.9486
	SRMDNF [52]	34.12	0.9254	30.04	0.8371	28.97	0.8025	27.57	0.8398	33.00	0.9403
4×	RDN [13]	34.71	0.9296	30.57	0.8468	29.26	0.8093	28.80	0.8653	34.13	0.9484
	CSFM (ours)	<b>34.76</b>	<b>0.9301</b>	<b>30.63</b>	<b>0.8477</b>	<b>29.30</b>	<b>0.8105</b>	<b>28.99</b>	<b>0.8681</b>	<b>34.52</b>	<b>0.9502</b>
	Bicubic	28.43	0.8109	26.00	0.7023	25.96	0.6678	23.14	0.6574	24.89	0.7875
	SRCNN [14]	30.48	0.8628	27.50	0.7513	26.90	0.7103	24.52	0.7226	27.63	0.8553
	FSRCNN [20]	30.70	0.8657	27.59	0.7535	26.96	0.7128	24.60	0.7258	27.85	0.8557
	VDSR [15]	31.35	0.8838	28.02	0.7678	27.29	0.7252	25.18	0.7525	28.87	0.8865
	LapSRN [27]	31.54	0.8866	28.09	0.7694	27.32	0.7264	25.21	0.7553	29.09	0.8893
	DRRN [16]	31.68	0.8888	28.21	0.7720	27.38	0.7284	25.44	0.7638	29.45	0.8946
	MemNet [19]	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630	29.64	0.8971
	IDN [31]	31.82	0.8903	28.25	0.7730	27.41	0.7297	25.41	0.7632	29.41	0.8936
5×	EDSR [12]	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.03	0.9158
	SRMDNF [52]	31.96	0.8925	28.35	0.7772	27.49	0.7337	25.68	0.7731	30.09	0.9024
	D-DBPN [29]	32.42	0.8977	28.76	0.7862	27.68	0.7393	26.38	0.7946	30.91	0.9137
	RDN [13]	32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028	31.00	0.9151
	CSFM (Ours)	<b>32.61</b>	<b>0.9000</b>	<b>28.87</b>	<b>0.7886</b>	<b>27.76</b>	<b>0.7432</b>	<b>26.78</b>	<b>0.8065</b>	<b>31.32</b>	<b>0.9183</b>

Figure 22: Quantitative result of CSFM (FMM=8 and CSAR=16).

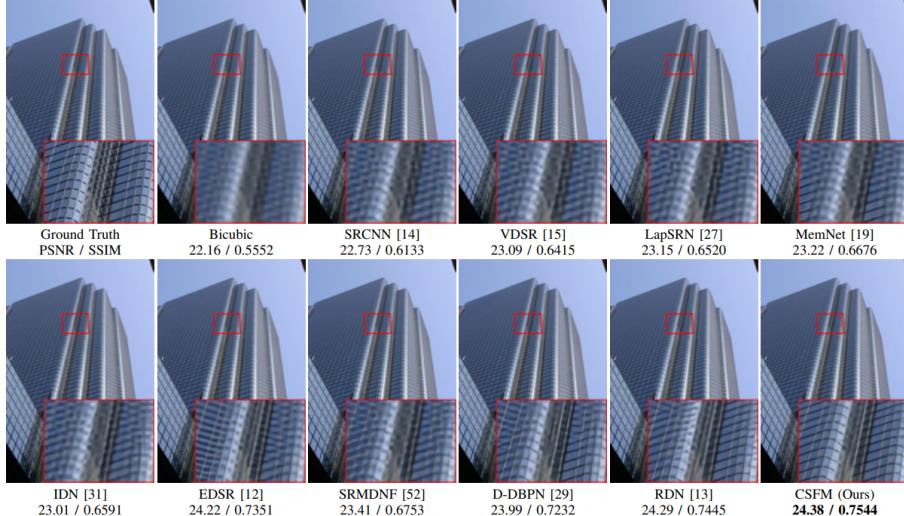


Fig. 12: Visual evaluation for a scale factor of 4x on the image "img074" from Urban100. The reconstructed grids produced by our CSFM network are more faithful and sharper than those by other methods.

Figure 23: Qualitative result of CSFM (FMM=8 and CSAR=16).

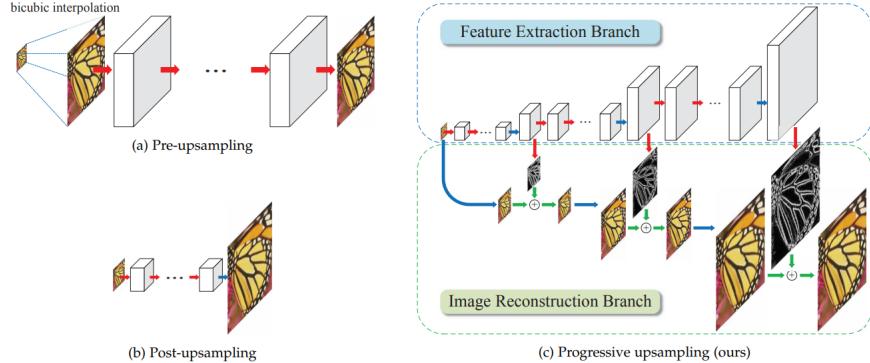


Fig. 1. **Comparisons of upsampling strategies in CNN-based SR algorithms.** Red arrows indicate **convolutional layers**, blue arrows indicate **transposed convolutions** (upsampling), and green arrows denote **elementwise addition** operators. (a) Pre-upsampling based approaches (e.g., SRCNN [9], VDSR [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 24: LapSRN architecture.

## 2.4 Deep Laplacian Pyramid Network (LapSRN[11]) and Multi-scale Deep Laplacian Pyramid Network (MS-LapSRN [12])

### 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.

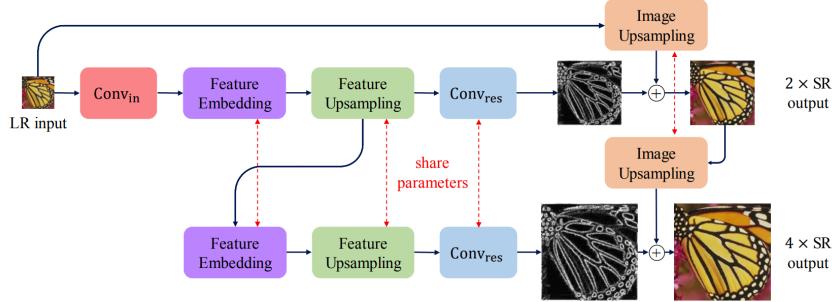


Fig. 3. **Detailed network architecture of the proposed LapSRN.** At each pyramid level, our model consists of a feature embedding sub-network for extracting non-linear features, transposed convolutional layers for upsampling feature maps and images, and a convolutional layer for predicting the sub-band 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 25: MS-LapSRN architecture.

#### 2.4.2 Architecture

In *LapSRN* [Figure 24] 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 25] the feature extractor (*Conv<sub>in</sub>*, Feature Embedding, Feature Upsampling, *Conv<sub>res</sub>*) 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 [6] [7] which contains distinct **D** convolutional layers.

The structure of the recursive block [Figure 26b] 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.

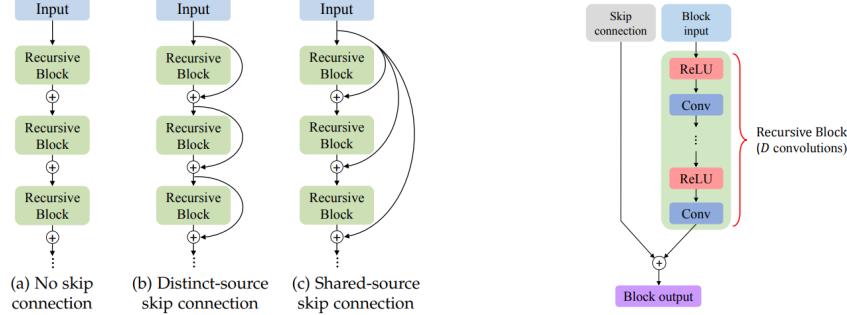


Fig. 4. **Local residual learning.** We explore three different ways of local skip connection in the feature embedding sub-network of the LapSRN for training deeper models.

(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**[12] since it's an evolution of LapSRN.

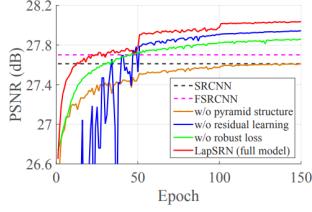


Fig. 6. **Convergence analysis.** We analyze the contributions of the pyramid structures, loss functions, and global residual learning by replacing each component with the one used in existing methods. Our full model converges faster and achieves better performance.

(a) Study on performance with different settings.

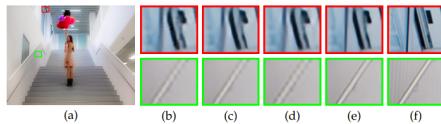


Fig. 7. **Contribution of different components in LapSRN.** (a) Ground truth HR image (b) without pyramid structure (c) without global residual learning (d) without robust loss (e) full model (f) HR patch.

(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	<b>28.04</b>	

(b) Ablation study on LapSRN.

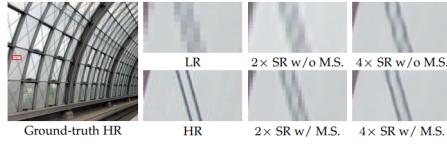


Fig. 8. **Contribution of multi-scale supervision (M.S.).** The multi-scale supervision guides the network training to progressively reconstruct the HR images and help reduce the spatial aliasing artifacts.

(d) Contribution of multi-scale supervision.

Figure 27: Performance studies done on MS-LapSRN

**Ablation studies on pyramid structure, global residual learning and loss** From the performance studies [Figure 27] 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 28] 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<sub>SS</sub>) achieves superior performance for deeper models.

Model	Depth	LapSRN <sub>NS</sub>	LapSRN <sub>DS</sub>	LapSRN <sub>SS</sub>
D5R2	24	25.16	25.22	<b>25.23</b>
D5R5	54	25.18	25.33	<b>25.34</b>
D5R8	84	25.26	25.33	<b>25.38</b>

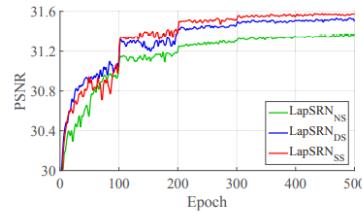


Fig. 9. **Comparisons of local residual learning.** We train our LapSRN-D5R5 model with three different local residual learning methods as described in Section 3.2.3 and evaluate on the SET5 for 4× SR.

Figure 28: 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	<b>27.32</b>	<b>25.21</b>
LapSRN <sub>NS</sub> -D10R1	407k	<b>27.32</b>	25.20
LapSRN <sub>NS</sub> -D5R2	222k	27.30	25.16
LapSRN <sub>NS</sub> -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	<b>25.31</b>
D2R20	112k	84	<b>27.37</b>	<b>25.31</b>
D4R3	185k	28	27.33	25.25
D4R6	185k	52	<b>27.37</b>	25.34
D4R10	185k	84	<b>27.37</b>	<b>25.35</b>
D5R2	222k	24	27.32	25.23
D5R5	222k	54	27.38	25.34
D5R8	222k	84	<b>27.39</b>	<b>25.38</b>
D10R1	407k	24	27.33	25.23
D10R2	407k	44	27.36	25.27
D10R4	407k	84	<b>27.38</b>	<b>25.36</b>

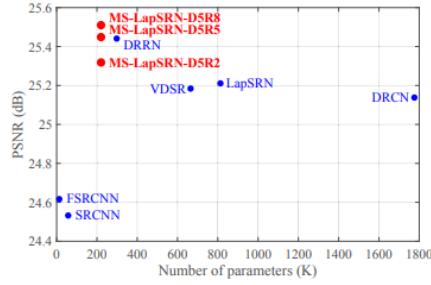


Fig. 15. **Number of network parameters versus performance.** The results are evaluated on the URBAN100 dataset for 4× SR. The proposed MS-LapSRN strides a balance between reconstruction accuracy and execution time.

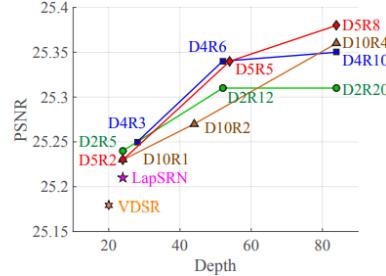


Fig. 10. **PSNR versus network depth.** We test the proposed model with different D and R on the URBAN100 dataset for 4× SR.

Figure 29: 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/IFC for 2x, 3x, 4x and 8x SR. Red and blue indicate the best and the second best performance, respectively. Both LapSRN<sup>+</sup> and the proposed MS-LapSRN do not use any 3x SR images for training. To generate the results of 3x SR, we first perform 4x SR on input LR images and then downsample the output to the target resolution.

Fig. 11. Visual comparison for 4x SR on the BSDS100, URBAN100 and MANGA109 dataset.



(1)  $\bigcap_{i=1}^k \text{Im}(f_i) = \{0\}$  if and only if  $f$  is surjective.

(a) SOTa results up to now



(c) Multi-scale results of MS-LapSRN

Figure 30: Quantitative and qualitative results of MS-LapSRN

**Studies on parameter sharing** From the parameters studies [Figure 29] it's possible to see the effectiveness of using recursive block which allow to have a deeper network with an increase of performance without too much parameters and also a faster inference time.

**Quantitative and qualitative results** From the quantitative and qualitative results [Figure 30] we can see that overall LapSRN performs better than state-of-the-art network, indeed it is able to reconstruct finer details thanks to the multi-scale training as we can see in the figure.

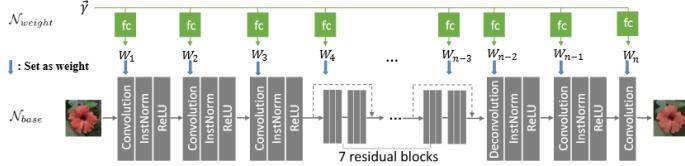


Figure 31: Parametrized Image Operator network.

## 2.5 Magnification-Arbitrary Network (MetaSR [10])

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 [11]) then there are also training cost and memory cost.

In order to overcame fixed integer problems [10] 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 31:

- $N_{base}$  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.

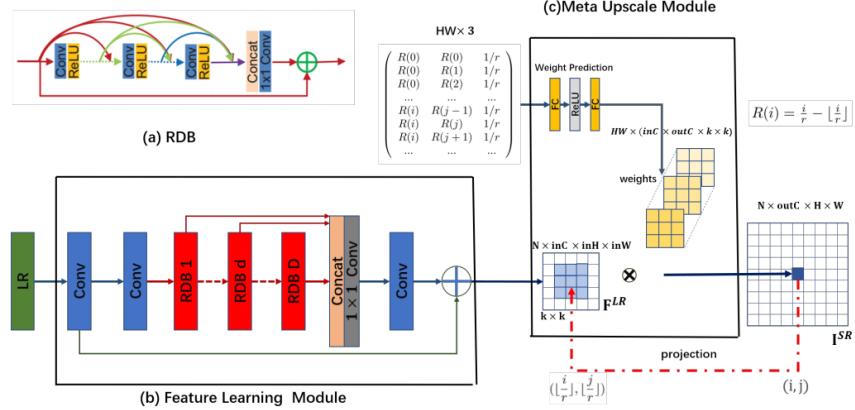


Figure 1. An instance of our Meta-SR based on RDN [36]. We also call the network Meta-RDN. (a) The Residual Dense Block proposed by RDN [36]. (b) The Feature Learning Module which generates the shared feature maps for arbitrary scale factor. (c) For each pixel on the SR image, we project it onto the 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 32: 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.

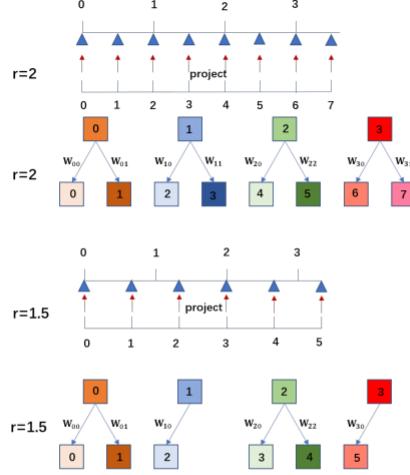


Figure 2. The schematic diagram for how to upscale the feature map with the non-integer scale factor  $r = 1.5$ . Here we only show the one-dimensional case for simplify.

Figure 33: Location projector: each pixel in the SR image (bottom row) has an unique pixel in the LR image (top row).

- **weight prediction** which predicts the weights of the kernel for each pixel  $W(i, j) = \phi(v_{ij}, \theta)$  where  $\phi$  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:

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

(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:

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

### 2.5.3 Results

**Results with decimal scales** From the results [Figure 34] 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( $x_1$ ) and EDSR( $x_1$ ): LR image is upscaled by  $r$  and processed by the network
- RDN( $x_k$ ) and EDSR( $x_k$ ): LR image is processed by the network which apply a upscaling by  $k$  then the HR image is downsampled 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.

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( <b>our</b> )	<b>42.82</b>	<b>40.40</b>	<b>38.28</b>	<b>36.95</b>	<b>35.86</b>	<b>34.90</b>	<b>34.13</b>	<b>33.45</b>	<b>32.86</b>	<b>32.35</b>
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( <b>our</b> )	<b>42.72</b>	<b>39.92</b>	<b>38.16</b>	<b>36.84</b>	<b>35.78</b>	<b>34.83</b>	<b>34.06</b>	<b>33.36</b>	<b>32.78</b>	<b>32.26</b>
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( <b>our</b> )	<b>31.82</b>	<b>31.41</b>	<b>31.06</b>	<b>30.62</b>	<b>30.45</b>	<b>30.13</b>	<b>29.82</b>	<b>29.67</b>	<b>29.40</b>	<b>29.30</b>
EDSR(x1)	31.69	31.29	<b>30.91</b>	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( <b>our</b> )	<b>31.73</b>	<b>31.31</b>	30.87	<b>30.60</b>	<b>30.40</b>	<b>30.09</b>	<b>29.83</b>	<b>29.61</b>	<b>29.34</b>	<b>29.22</b>
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	<b>28.09</b>	27.84	27.61	27.51
RDN(x4)	<b>28.89</b>	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( <b>our</b> )	28.87	<b>28.79</b>	<b>28.68</b>	<b>28.54</b>	<b>28.32</b>	<b>28.27</b>	28.04	<b>27.92</b>	<b>27.82</b>	<b>27.75</b>
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( <b>our</b> )	<b>28.95</b>	<b>28.82</b>	<b>28.63</b>	<b>28.48</b>	<b>28.27</b>	<b>28.21</b>	<b>27.98</b>	<b>27.86</b>	<b>27.75</b>	<b>27.67</b>

Figure 34: Results on Meta-SR

**Comparison between RDN and Meta-RDN** Overall Meta-SR performs better than RDN.



Figure 35: Meta-SR applied for different scales to a single image.

Methods	Metric	Set14			B100			Manga109			DIV2K		
		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	<b>30.57</b>	28.81	32.34	29.26	27.72	<b>39.18</b>	34.13	31.00	35.17	31.39	29.34
	SSIM	0.9212	<b>0.8469</b>	0.7871	0.9017	0.8093	0.7419	0.9780	<b>0.9484</b>	0.9151	0.9483	0.8931	0.8446
Meta-RDN	PSNR	<b>34.04</b>	30.55	<b>28.84</b>	<b>32.35</b>	<b>29.30</b>	<b>27.75</b>	39.18	<b>34.14</b>	<b>31.03</b>	<b>35.18</b>	<b>31.42</b>	<b>29.36</b>
	SSIM	<b>0.9213</b>	0.8466	<b>0.7872</b>	<b>0.9019</b>	<b>0.8096</b>	<b>0.7423</b>	<b>0.9782</b>	0.9483	<b>0.9154</b>	<b>0.9484</b>	<b>0.8935</b>	<b>0.8448</b>

Figure 36: Meta-SR versus RDN.

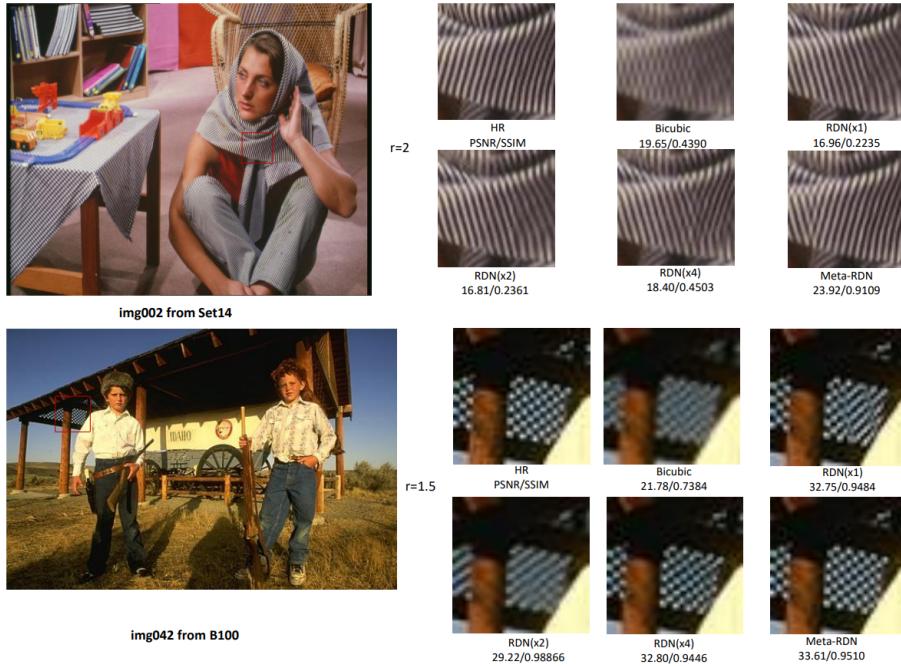


Figure 37: The visual comparison with four baselines. Our MetaRDN has better performance.

## 2.6 Extra

### 2.6.1 The Laplacian Pyramid [14]

In order to compress an image reducing the pixel correlations the *Laplacian Pyramid* was created: subtracting an image with its low-bass band filtered image (using a hand-crafted kernel) it's possible to reduce the variance and the entropy of the image itself; in this way less bits are necessary for representing the image.

The name is due to the similar result obtained applying a laplacian operator to an image.

The kernel used is a 5x5 gaussian-like kernel:

- **separable:**  $W(m, n) = \tilde{w}(m) \cdot \tilde{w}(n)$
- **symmetric:**  $\tilde{w}(i) = w(-i), i \in [1, 2]$ , where  $w(0) = a, w(1) = w(-1) = b, w(2) = w(-2) = c$
- **normalized:**  $a + 2c + 2b = 1$
- **equal contribution:**  $a + 2c = 2b$

which leads to:

$$W = \begin{cases} w(0) = a \\ w(1) = w(-2) = \frac{1}{4} \\ w(2) = w(-2) = \frac{1}{4} - \frac{a}{2} \end{cases}$$

Given  $\alpha$  different kernel are created:

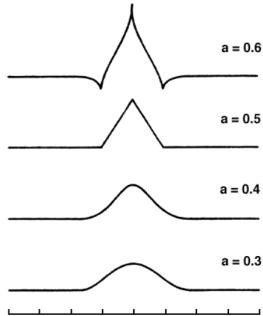


Fig. 3. The shape of the equivalent weighting function depends on the choice of parameter  $a$ . For  $a = 0.5$ , the function is triangular; for  $a = 0.4$  it is Gaussian-like, and for  $a = 0.3$  it is broader than Gaussian. For  $a = 0.6$  the function is trimodal.

Figure 38: Kernels as function of  $\alpha$

Then, first, a **Gaussian pyramid** is created convolving the kernel with the image *reducing* each time the spatial dimension then each low-bass band filtered image is subtracted with the low-pass band filtered image in the lower level

(after *expanding* it) creating a **Laplacian pyramid** where **high frequency information** are highlighted at different scales.

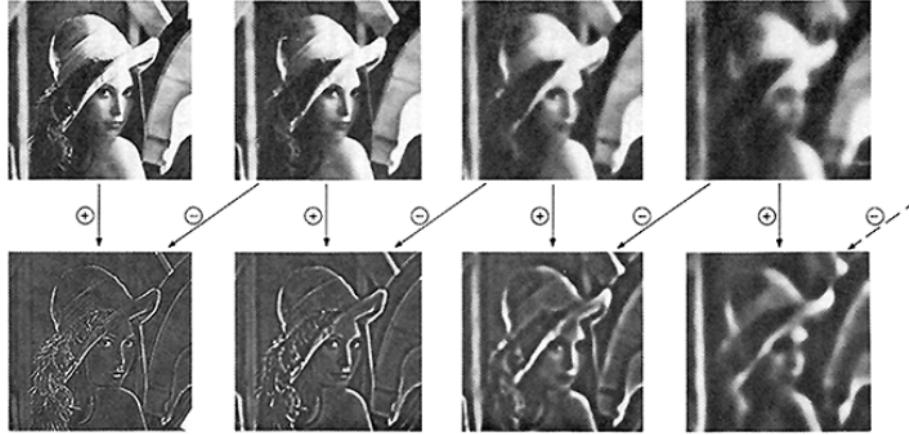


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

Figure 39: Laplacian pyramid example. The original image is the one on the left. Each image is REDUCED and EXPANDED for creating the Laplacian Pyramid.

The image can be reconstructed starting from the bottom level and expanding and summing two consecutive Laplacian images:

$$g_l = L_l + \text{EXPAND}(g_{l+1})$$

Further studies on the entropy and quantization allowed researcher to reduce much more the quantity of information necessary for representing the image with a small loss in quality.

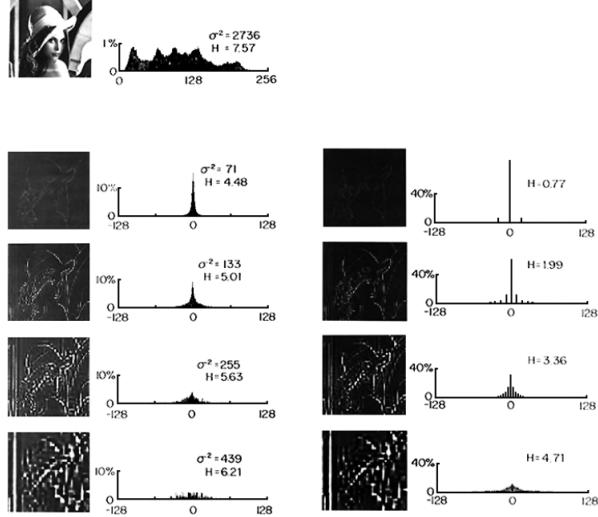


FIG 6. The distribution of pixel gray level values at various stages of the encoding process. The histogram of the original image is given in (a). (b)-(e) give histograms at levels 0-3 of the Laplacian pyramid with generating parameter  $a=0.6$ . Histograms following quantization at each level are shown in (f)-(i). Note that pixel values in the Laplacian pyramid are concentrated near zero, permitting data compression through shortened and variable length code words. Substantial further reduction is realized through quantization (particularly at low pyramid levels) and reduced sample density (particularly at high pyramid levels).

(a) Study on entropy and quantization on the Laplacian Pyramid.

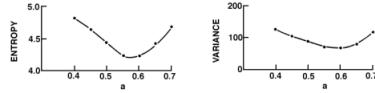


Fig 7. Entropy and variance of pixel values in Laplacian pyramid level 0 as a function of the parameter "a" for the "Lady" image. Greatest reduction is obtained for  $a \approx 0.6$  This estimate of the optimal "a" was also obtained at other pyramid levels and for other images.

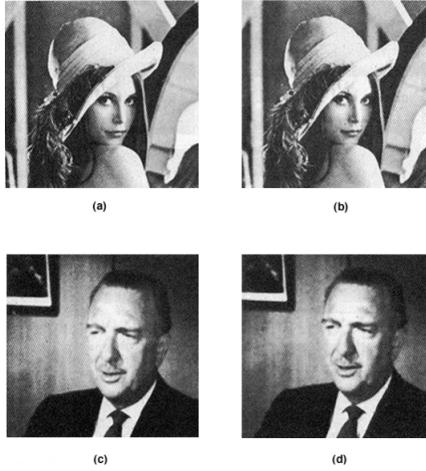
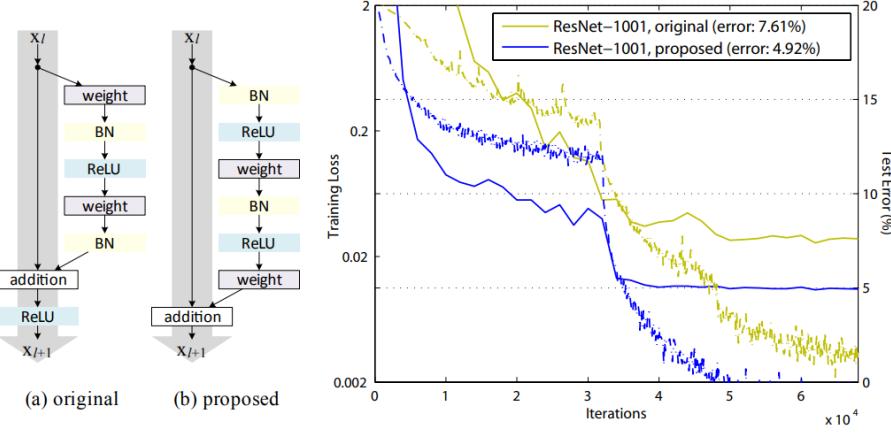


Fig 8. Examples of image data compression using the Laplacian Pyramid code. (a) and (c) give the original "Lady" and "Walter" images, while (b) and (d) give their encoded versions of the data rates are 1.58 and 0.73 bits/pixel for "Lady" and "Walter," respectively. The corresponding mean square errors were 0.88 percent and 0.43 percent, respectively.

- (b) Examples with quantization using as width of bins a value such that there is a small degradation on the perception of the image at distance five times greater than the image dimensions.

### 2.6.2 Identity mapping as output of the residual block [15]



**Figure 1. Left:** (a) original Residual Unit in [1]; (b) proposed Residual Unit. The grey arrows indicate the easiest paths for the information to propagate, corresponding to the additive term “ $x_l$ ” in Eqn.(4) (forward propagation) and the additive term “1” in Eqn.(5) (backward propagation). **Right:** training curves on CIFAR-10 of **1001-layer** ResNets. Solid lines denote test error (y-axis on the right), and dashed lines denote training loss (y-axis on the left). The proposed unit makes ResNet-1001 easier to train.

Figure 41: The results of using a clean path in the identity.

The result [Figure 41] shows that a **clean information path** from the input and the output (gray line in the figure) allows to ease the training improving the performance.

In order to do so an **identity mapping** should be used in the output of the residual block therefore the researcher proposed a new residual block where the input is **pre-activated**.

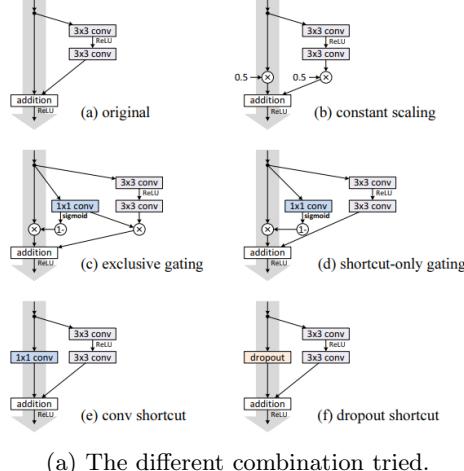
A *Residual unit* can be formalized in this way: 
$$\begin{cases} y_l = h(x_l) + F(x_l, W_l) \\ x_{l+1} = f(y_l) \end{cases} \quad \text{where}$$
  $x_l$  and  $x_{l+1}$  are, respectively, the input and the output of the  $l$ -th residual unit,  $y_l$  is the output of the sum between  $h$ , identity, and  $F$  the residual, and  $f$  is the *ReLU*.

Using an identity mapping in the output then  $x_{l+1} = y_l = x_l + F(x_l, W_l)$  which recursively lead to  $X_L = x_l + \sum_i=1^{L-1} F(x_i, w_i)$ : any deeper unit  $L$  is the sum of the input of a shallow unit  $l$  and the residual of the unit between  $l$  and  $L$ .

An intuition of why this is used in Section 2.4 and DRRN[7] can be seen in this results: since the skip connection allow to flow the real input (the LR image) than the network is able to take apart low frequency information better.

The backpropagation lead to:  $\frac{\partial \epsilon}{\partial x_l} = \frac{\partial \epsilon}{\partial x_L} \frac{\partial x_L}{\partial x_l} = \frac{\partial \epsilon}{\partial x_L} \left( 1 + \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} F(x_i, W_i) \right)$  which is never 0 because is unlikely that  $\frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} F(x_i, W_i)$  is always  $-1$  for all minibatches.

**Studies done on different residual block** In order to prove the mathematical claims they experimented with different combination whose result are worse than the proposed residual block.



(a) The different combination tried.

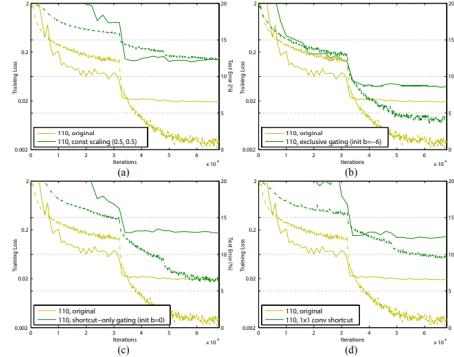


Figure 3. Training curves on CIFAR-10 of various shortcuts. Solid lines denote test error (y-axis on the right), and dashed lines denote training loss (y-axis on the left).

(b) The result of the different combination tried.

**Studies on the effectiveness of the pre-activation** The researchers experimentally prove the correctness of the pre-activation residual block.

**Table 2.** Classification error (%) on the CIFAR-10 test set using different activation functions.

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
<b>full pre-activation</b>	Fig. 4(e)	<b>6.37</b>	<b>5.46</b>

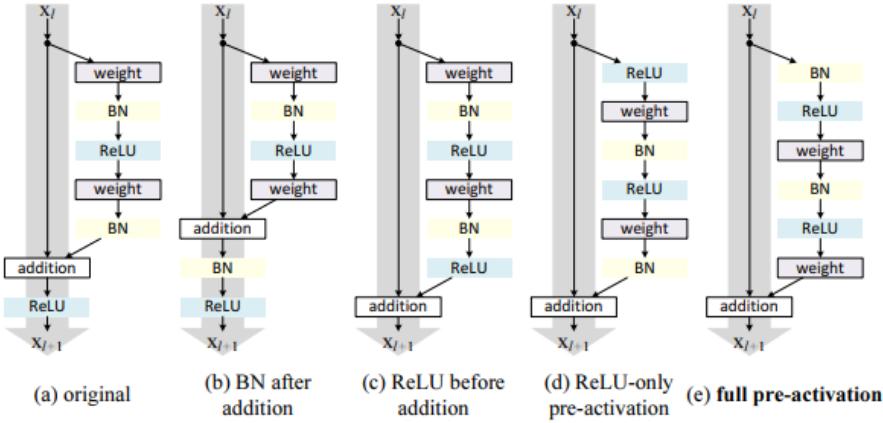


Figure 43: Different combinations of disposition of batch normalization and ReLU

### 2.6.3 Pixel Shuffle: sub-pixel convolution [13]

The **Pixel Shuffle** rearrange the features  $C \cdot r^2 \times H \times W$  learned into the LR space into  $C \times 2 \cdot H \times 2 \cdot W$  activations learned into the HR space.

In [17], the authors proved that transposed convolution and subpixel convolution return the same result and convolving the features map with a convolution  $(r^2 \cdot C_{out}, C_{in}, k_H, k_W)$  where  $r$  is the upscale factor and then rearrange the activations in order to have an activation whose size is  $(r \cdot H, r \cdot W)$  is the same as apply a transposed convolution  $(C_{out}, C_{in}, r \cdot k_H, r \cdot k_W)$  ( fractional strided convolution where the input is modified inserting zeros in between and padded in order to have an output upscaled with respect the input)

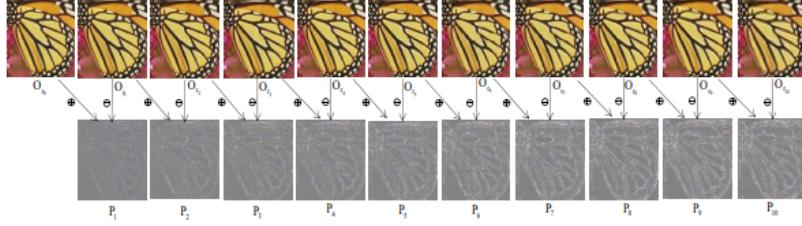


Fig. 2: The Laplacian Frequency Representation has 11 Laplacian pyramid levels ( $O_{r_0}, \dots, O_{r_{10}}$ ), with 10 phases in the scale range of  $(1, 2]$  ( $P_1, \dots, P_{10}$ ). Each phase represents the difference between the successive Laplacian pyramid levels.

Figure 44: Laplacian Frequency Representation using decimal scales

### 3 Any-Scale Deep Network (ASDN[1])

**ASDN** and *Meta-SR/10*[Section 2.5] are few among SR networks whose objective is to achieve SR images using decimal scales.

In order to do so *Meta-SR* used a *meta-upscaling module* which learns the weights used for convolving features extracted by a LR image given the scale; therefore *Meta-SR* is able to predict the SR image only for scales used for training.

For overcoming this and the impossibility to train a network for all possible decimal scales, *ASDN* eases the task using a **Laplacian Frequency Representation**: exploiting the Laplacian Framework with decimal scales (seen in Section 2.4) the SR image is reconstructed interpolating two nearest neighbors of the decimal scale in the laplacian pyramid.

#### 3.1 Laplacian Frequency Representation

Each level learns the high frequency image at the specific scale which is defined as:

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

then given any decimal scale  $r$  the nearest neighbor is chosen with:

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

and the weights used for interpolating are:

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

therefore the **SR image** is:

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

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

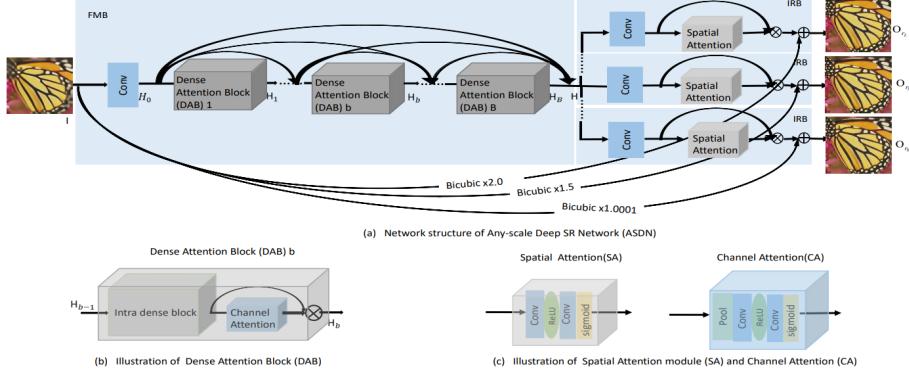


Figure 45: ASDN architecture

The pyramid is named *Laplacian Frequency Representation (LFR)* because at each level a phase is computed which represents the high frequency information of the input image as if we are applying a laplacian operator, indeed looking at Figure 44 what we have is a sort of edge detector.

### 3.2 Recursive deployment

In order to avoid to train the network for all possible decimal scales the *Laplacian Frequency Representation* is fixed to a range therefore in order to be able to upscale decimal scales greater than the greatest scale in the pyramid the network is deployed recursively. In general a large scale  $R$  can be defined as  $R = r^N$  where  $N$  is the number of recursions.

The best scales range is found to be  $(1, 2]$  therefore the network should be deployed  $N - 1$  times with  $r = 2$  and once with  $r = \frac{R}{2^{N-1}}$  where  $N = \lceil \log_2 R \rceil$

### 3.3 Architecture

Looking at [Figure 45], ASDN has:

- a shallow features extractor
- a deeper features extractor, **Feature Mapping Branch (FMB)**
- at each level in the *LFR* the features are reconstructed using the **Image Reconstruction Branch (IRB)**
- the *FMB* is shared among all *IRBs*

The *ASDN* is the aggregator of all previous studies:

- the **bi-dense connections** allow to reuse all features within the network, maintain long-term information and ease the training avoiding vanishing/exploding gradient (seen in Section 2.1)

- the IDBs taken directly from Section 2.1 allow to maintain short-term information focusing the training on most important features thanks of the **channel attention**(as seen in Section 2.2 and Section 2.3)
- the IRB project the features in the image space using a 1x1 convolution with  $C = 3$  (or 1 in case of gray-scale images) and then the reconstruction is focused on particular region where high-frequency information are laying due to the **spatial attention** mechanism (as seen in Section 2.3)

### 3.4 FSDN

*Fixed-scale deep network (FSDN)* is a network similar to ASDN, with the only exception of a transposed convolution at the beginning of each IRB, which is fine tuned to a specific scale.

### 3.5 Results

#### 3.5.1 Efficiency of the Laplacian Frequency Representation and density study.

Looking at [Figure 46] the Laplacian Frequency Representation (network trained using 11 IRBs: EDSR-Conv, RDN-Conv, ASDN) has the same performances of networks trained using pre-upsampling method (EDSR-100, RDN-100, ASDN-100 on 100 scales between 1 and 2).

We can see that the denser the pyramid the better but at some point it saturate and no benefit are achieved (ASDN- $x$  where  $x$  is the number of phases in the LFR so  $L = x + 1$ .)

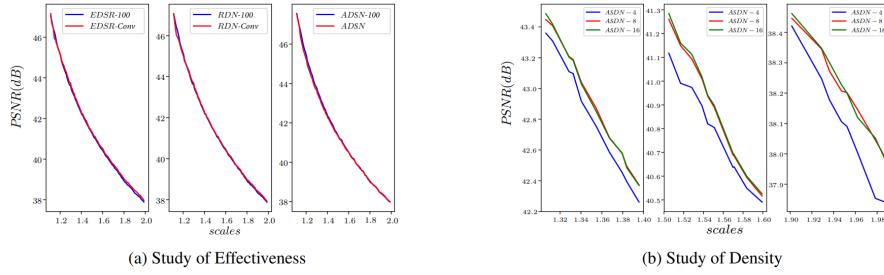


Fig. 8: Study of Laplacian Frequency Representation

Figure 46: Effectiveness and density of LFR.

#### 3.5.2 Efficiency of the recursive deployment and optimal N and r

The direct deployment uses networks trained with images upscaled directly to  $x2, x3, x4$  instead recursive deployment uses networks trained twice using  $x\sqrt{2}, x\sqrt{3}, x\sqrt{4}$ .

From [Figure 47] we can see that the difference in performance is noticeable with lower scales and decrease with it and using the greatest scale at the beginning lead to better performance.

Methods	Direct deployment			Recursive deployment		
	$\times 2$	$\times 3$	$\times 4$	$\times 2$	$\times 3$	$\times 4$
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  $\times 2$ ,  $\times 3$ ,  $\times 4$

(a) Study of the effectiveness of the recursive deployment.

Scale(R)	Recursion(N)	UpscaleRatio(r)	PSNR
$3\times$	2	1.732, 1.732	34.43
		1.500, 2.000	34.19
		2.000, 1.500	<b>34.48</b>
	3	1.442, 1.442, 1.442	33.18
$4\times$	2	2.000, 2.000	<b>32.27</b>
		1.587, 1.587, 1.587	31.96
	3	1.800, 1.800, 1.234	32.16
		2.000, 1.800, 1.100	32.24

(b) Optimal N and r for the recursive deployment.

Figure 47: Recursive deployment studies.

### 3.5.3 Ablation studies and performance versus parameters

Every component inside ADSN is important for achieving better performance as shown in Figure 48a.

The network is able to achieve state-of-the-art performance with a small amount of parameters (22M).

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	<b>38.01</b>

(a) Ablation studies on channel attention (CA), spatial attention (SA) and global skip connection (SC).

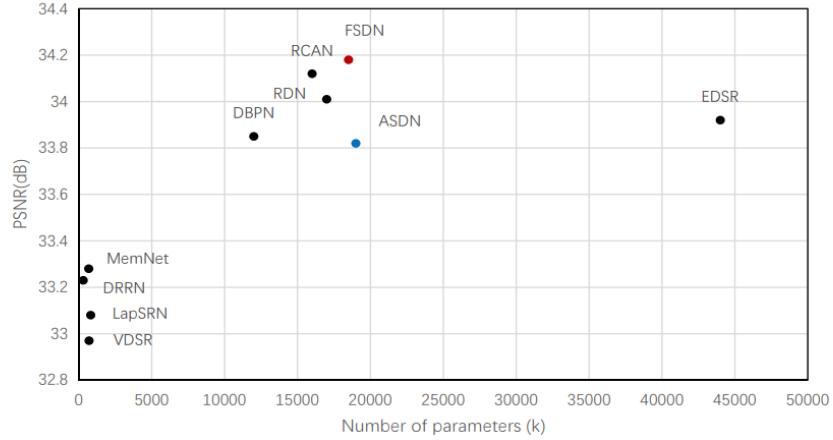


Fig. 9: Performance vs number of parameters. The results are evaluated with Set14 for 2× enlargement. Red indicates the best performance, and blue indicates the best performance among predefined upsampling methods

(b) PSNR versus number of parameters.

### 3.5.4 Quantitative and qualitative results

**Fixed scales** Overall ASDN performs better than any predefined upsampling method and FSDN has the best performance (this highlight the fact that having learnable upscaling method allow to achieve better performance).

Sometimes ASDN performs worse because some network are trained specifically for a particular scales with the particular dataset or it is worse than RCAN because has many channel attention modules that allow to improve the reconstruction.

Table 1: Quantitative evaluation of state-of-the-art SR algorithms. We report the average PSNR/SSIM for  $2\times$ ,  $3\times$ ,  $4\times$  and  $8\times$  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								
$2\times$	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)	<b>38.12</b>	<b>0.961</b>	<b>33.82</b>	<b>0.919</b>	<b>32.30</b>	<b>0.901</b>	<b>32.47</b>	<b>0.931</b>	<b>39.16</b>	<b>0.978</b>
$3\times$	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	<b>33.34</b>	<b>0.938</b>	39.44	0.979
	FSDN(ours)	<b>38.27</b>	<b>0.961</b>	<b>34.18</b>	<b>0.923</b>	<b>32.41</b>	<b>0.903</b>	33.13	0.937	<b>39.49</b>	<b>0.979</b>
	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
$4\times$	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)	<b>34.48</b>	<b>0.928</b>	<b>30.35</b>	<b>0.843</b>	<b>29.18</b>	<b>0.808</b>	<b>28.45</b>	<b>0.858</b>	<b>33.87</b>	<b>0.947</b>
	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	<b>30.65</b>	0.848	29.32	0.811	<b>29.09</b>	<b>0.870</b>	34.44	0.949
$8\times$	FSDN(ours)	<b>34.75</b>	<b>0.930</b>	30.63	<b>0.848</b>	<b>29.33</b>	<b>0.811</b>	28.98	0.868	<b>34.53</b>	<b>0.950</b>
	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)	<b>32.27</b>	<b>0.896</b>	<b>28.66</b>	<b>0.784</b>	<b>27.65</b>	<b>0.740</b>	<b>26.27</b>	<b>0.792</b>	<b>30.91</b>	<b>0.913</b>
$8\times$	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	<b>26.82</b>	<b>0.809</b>	31.22	0.917
	FSDN(ours)	<b>32.63</b>	<b>0.900</b>	<b>28.89</b>	<b>0.789</b>	<b>27.79</b>	<b>0.744</b>	26.79	0.807	<b>31.44</b>	<b>0.919</b>
	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
$8\times$	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)	<b>27.02</b>	<b>0.776</b>	<b>24.99</b>	<b>0.641</b>	<b>24.82</b>	<b>0.600</b>	<b>22.57</b>	<b>0.620</b>	<b>24.73</b>	<b>0.748</b>
	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	<b>0.606</b>	<b>23.00</b>	<b>0.645</b>	25.24	0.803
	FSDN(ours)	<b>27.33</b>	<b>0.789</b>	<b>25.24</b>	<b>0.651</b>	<b>24.98</b>	0.604	22.90	0.638	<b>25.24</b>	<b>0.803</b>

Figure 49: Comparison of quantitative results using fixed scales.

**Any-scale** ASDN performs better than state-of-the-art for upsampling images using decimal scales used or not for training.

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	<b>42.92</b>	40.11	<b>38.33</b>	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	<b>32.78</b>
Meta-RDN [8]	42.82	<b>40.40</b>	38.28	<b>36.95</b>	<b>35.86</b>	<b>34.90</b>	<b>34.13</b>	<b>33.45</b>	<b>32.86</b>
ASDN(ours)	<b>43.05</b>	<b>40.24</b>	<b>38.42</b>	<b>37.02</b>	<b>35.87</b>	<b>34.92</b>	<b>34.14</b>	<b>33.46</b>	<b>32.86</b>
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	<b>26.01</b>	<b>24.78</b>	<b>23.65</b>	<b>22.63</b>	<b>21.55</b>	<b>20.53</b>
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	<b>27.67</b>	-	-	-	-	-	-
Meta-RDN [8]	<b>32.35</b>	<b>29.67</b>	<b>27.75</b>	-	-	-	-	-	-
ASDN(ours)	<b>32.30</b>	<b>29.63</b>	27.65	<b>26.07</b>	<b>24.85</b>	<b>23.70</b>	<b>22.66</b>	<b>21.59</b>	<b>20.55</b>

(a) Comparison of quantitative results using decimal scales.

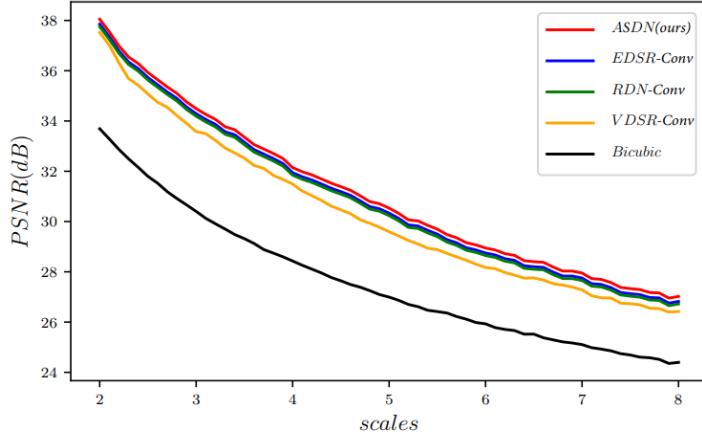


Fig. 4: PSNR comparison of ASDN with other works within the continuous scale range ( $\times 2, \times 8$ ] on Set5

(b) Comparison of quantitative results using decimal scales between 2 and 8.

**Qualitative results** The visual quality of ASDN is similar to state-of-the-art and FSDN outperforms state-of-the-art networks.



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

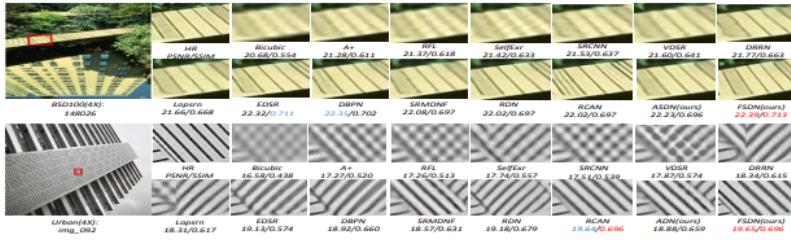
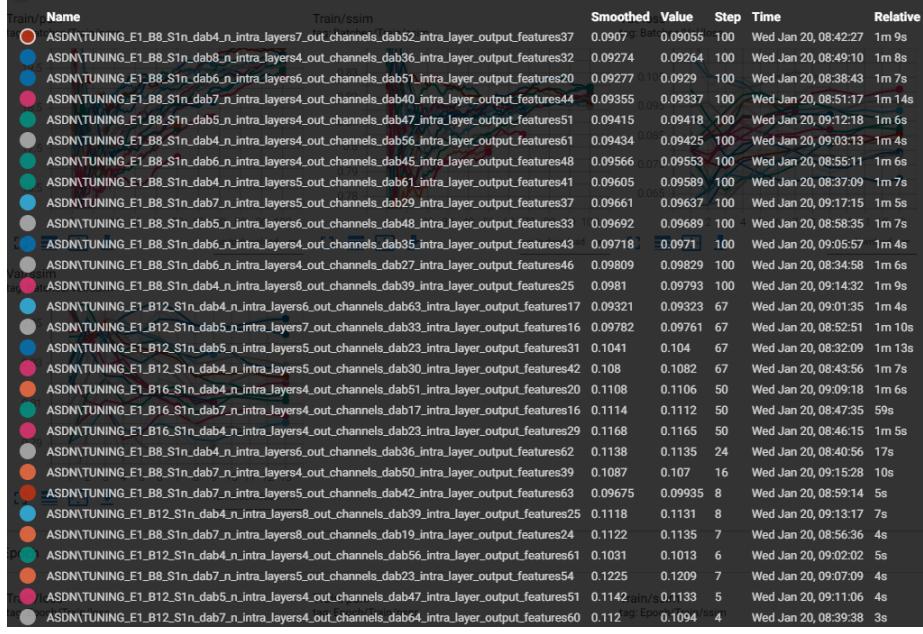


Fig. 6: Qualitative comparisons of our models with other works on  $\times 4$  super-resolution. Red indicates the best performance, and blue indicates the second best



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

Figure 51: Qualitative results of ASDN and FSDN on different datasets.



(a) Training time of different random configurations. There are many other configurations missing due to *out of memory* error.



(b) Comparison between the training time of 1 epoch of the default configuration, using the trick and not using the trick.

Figure 52: Trining time of 1 epoch. The name contains: E is *epoch*, B is *batch size*, S is *save checkpoint* and the network parameters.

## 4 Project

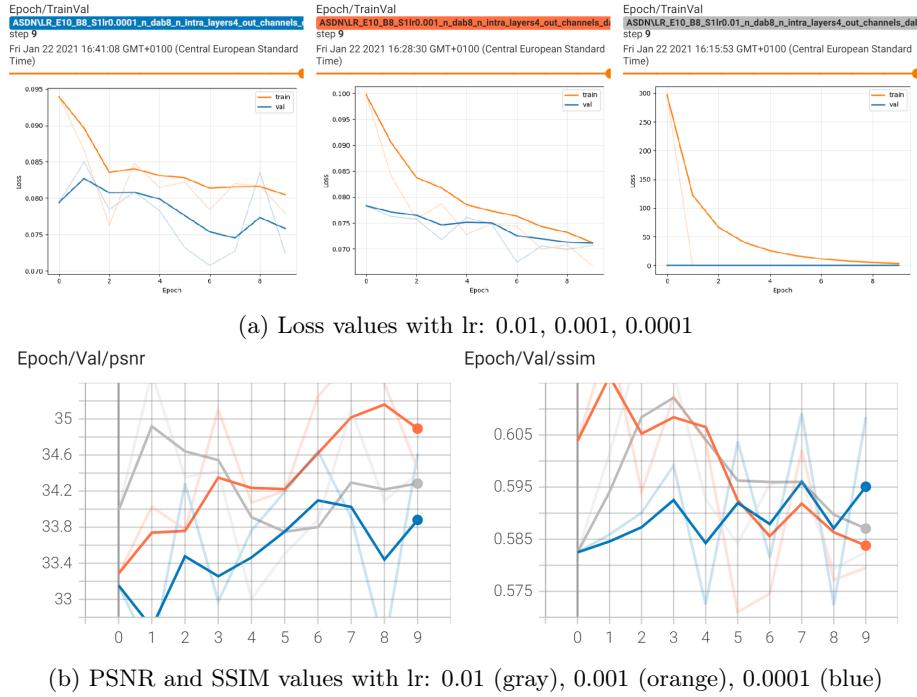
### 4.1 Hardware

The following hardware is used:

- GTX1060 (6 GB)

### 4.2 Architecture configuration

**ASDN** is using a bi-dense connection this means a lot of memory is used for storing intermediate activations as well as gradients: the default configuration, number of dense attention block 16 and number of intra connected layers 8, is not suitable for my computer.



In order to train the default configuration an implementation trick can be used (**save\_checkpoint**<sup>1</sup>): avoid to store gradient and activations during the forward pass and recompute them during the backward pass; but this increase the training time of a **single batch** as we can see from Figure 52b

Among many configurations tried [Figure 52a] the final configuration chosen uses

DABs	8
IDBs	4
Features DAB	32
Features IDB	32

### 4.3 The initial LR

The initial learning rate is set to **0.001** based on the experiments done.

---

<sup>1</sup>PyTorch checkpoint

## 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] 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.
- [3] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *CoRR*, vol. abs/1501.00092, 2015.
- [4] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image restoration,” *CoRR*, vol. abs/1812.10477, 2018.
- [5] Y. Wang, J. Shen, and J. Zhang, “Deep bi-dense networks for image super-resolution,” *CoRR*, vol. abs/1810.04873, 2018.
- [6] J. Kim, J. K. Lee, and K. M. Lee, “Deeply-recursive convolutional network for image super-resolution,” *CoRR*, vol. abs/1511.04491, 2015.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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.
- [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.
- [17] W. Shi, J. Caballero, L. Theis, F. Huszar, A. P. Aitken, C. Ledig, and Z. Wang, “Is the deconvolution layer the same as a convolutional layer?,” *CoRR*, vol. abs/1609.07009, 2016.