Exploiting Reflectional and Rotational Invariance in Single Image Superresolution

Simon Donn, Laurens Meeus, Hiep Quang Luong, Bart Goossens, Wilfried Philips
Ghent University - TELIN - IPI
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
Simon.Donne@UGent.be

Abstract

Stationarity of reconstruction problems is the crux to enabling convolutional neural networks for many image processing tasks: the output estimate for a pixel is generally not dependent on its location within the image but only on its immediate neighbourhood. We expect other invariances, too. For most pixel-processing tasks, rigid transformations should commute with the processing: a rigid transformation of the input should result in that same transformation of the output. In existing literature this is taken into account indirectly by augmenting the training set: reflected and rotated versions of the inputs are also fed to the network when optimizing the network weights. In contrast, we enforce this invariance through the network design. Because of the encompassing nature of the proposed architecture, it can directly enhance existing CNN-based algorithms. We show how it can be applied to SRCNN and FSRCNN both, speeding up convergence in the initial training phase, and improving performance both for pretrained weights and after finetuning[1].

1. Introduction

Training a (deep) neural networks involves large, annotated, datasets. Because the networks comprise so many parameters that need to be optimized , overfitting is a very real problem – hence the need for large datasets. As such datasets are time-consuming and costly to create, the datasets are often synthetically augmented in various ways.

In pixel-wise image processing stationarity of the processing, i.e. locational invariance, plays an important role. When performing such tasks as segmentation, denoising, or superresolution, the outputs of the network are assumed to only depend on a small neighbourhood of the respective locations and not on the location within the image.

[1] Tensorflow training and test code is made publically available at http://telin.ugent.be/~sdonn/code/FSRCNN_SEF.zip.

Figure 1. There are eight rigid transformations of the input image that respect the sampling grid. Applying any of these to the input we expect the same transformation of the output: the reconstruction should commute with such transformations.

This is the rationale behind convolutional neural networks. Yet for many applications, a.o. single-image superresolution, additional invariances exist. Specifically, we propose a high-level network architecture that exploits the invariance to rigid transformations. We use (eight) rigid transforms that do not require resampling the pixel grid as these require no sampling-theoretic considerations and prove to provide plenty of additional constraint to the network, see Figure 1. Because this is done inside the network rather than by augmenting the training set, we not only lower the number of parameters but also introduce an additional averaging step between several estimates.

The proposed approach is applied to superresolution in two scenarios: a neural network that departs from the bicubic upsampling of the input image (SRCNN [3]), and a network that starts from the low-resolution input (FSRCNN [4]). In the first case, each input pixel corresponds with one output pixel; in the latter case, we need to infer multiple output pixel values per input pixel. We show for both cases that exploiting the reflectional and rotational invariance inside the network results in both faster training and better performance. In the latter case we also show how the network does not need to infer all F^2 subpixels per pixel location but only a small subset of these.

Section 2 discusses existing approaches for single-image superresolution and the role of invariances. In Section 3 we outline the approach in detail, explicitly handling the cases of single-image superresolution by factors 2, 3 and 4. In Section 4 we apply our proposed method to the existing architectures for single-image superresolution and show its impact on the training speed and final performance of the networks. The networks are trained and validated using the NTIRE17 dataset, originally provided for the CVPR NTIRE 2017 workshop, as well as the established Set5 and Set14 datasets.

2. Related Work

Broadly, single image superresolution techniques can be classified in three groups: interpolation methods, learning-based methods and reconstruction methods. Interpolation methods are perhaps the most well known: nearest neighbour, bicubic, sinc-based (e.g. Lanczos), ... They are simple to implement, but do not yield very good results due to interpolation artifacts such as blur, staircasing and ringing.

Reconstruction methods enforce prior knowledge about the output while requiring the reconstruction to be consistent with the low-resolution input [5, 7, 12, 2]. Wavelet methods [1, 16, 6] also belong in the last category.

Learning-based methods match parts of the low-resolution input image with entries in a dictionary, translating them into a high-resolution patch. The dictionary is learned in a training phase from a training dataset.

It is clear that deep learning methods also belong firmly in this category. Several authors have explored this application of neural networks [14, 3, 4, 9] and have achieved significant boosts over previous state of the art. Our novel approach shows how a well-chosen extension to an existing network (that exploits the reflectional and rotational invariances) results in faster convergence and better performance. Wang et al. showed how the domain expertise of conventional sparse-coding based methods can also result in good performances with much smaller network sizes [15].

As far as the deep learning techniques are concerned, there are two main approaches to take. Either we upsample the input image before feeding it to the network [8, 14, 3, 9, 15] (through a fast but rough method, e.g., interpolation-based methods), or we simply pass the low-resolution input to the network and let it learn the optimal upsampling during training [4]. We illustrate that both benefit from our proposed approach.

Timofte et al. have recently discussed several ways to improve example-based single image super resolution [13]. While data augmentation shines on the first spot of their list, rigid transformation invariance is not mentioned. Instead, they raise scale invariance: image self-similarity means that information from multiple scales can be exploited during reconstruction.

Perez et al., on the other hand, explicitly mention the invariance to rigid transformations [13]. Interestingly, their approach is conceptually inverse to ours: by applying patch preprocessing and their proposed Symmetry-Collapsing Transform they drastically lower the manifold of possible patches. As a result, they need smaller dictionaries which speeds up training and reconstruction. Contrastingly we process each patch multiple times after explicitly applying these transforms, after which we treat the various reconstructions as different estimates for the pixel values and average them. In the case of the FSRCNN-based network this results in a reduction of the number of outputs per patch, also implying the possibility for smaller dictionaries as they need to contain less information.

3. Proposed Approach

We apply the proposed approach to two different scenarios. First, we take a look at a superresolution network that uses an upsampled version of the low-resolution image as its input. Correspondingly, it needs only output one pixel value per input pixel. Secondly, we also exploit the discussed invariances in a network that takes the low-resolution image as input. Typical networks for this scenario need to output $F \times F$ pixel values per input pixel, where F is the upsampling factor. Our proposed approach, however, elegantly reduces the required number of output. For upsampling with a factor $F = 2$, the proposed network need only output a single pixel value per input pixel.

3.1. Superresolution and rigid transformations

When applying the network to a rotated and/or reflected version $H(I)$ of the original input I we expect the resulting estimate $\mathcal{S}(H(I)) = (\mathcal{S} \circ H)(I)$ to be that same rotation and/or reflection of the original estimate $H(\mathcal{S}(I)) = (H \circ \mathcal{S})(I)$. In other words, we assume that the superresolution operation commutes with the rigid transformation of images. The only restriction we place on the transformation of the images is that it does not require resampling of the pixel grid. More general transformations, whose sampling locations do not coincide with the input grid, incur larger computational penalties and may, in extreme cases, introduce artifacts themselves. For this reason, we do only consider those rotations that do not require a resampling of the pixel grid.

3.2. SRCNN variants

By first bicubicaly upsampling the low-resolution input image it is no longer necessary to perform this upsampling within the network, which can now focus on resolving the artifacts introduced by the bicubic upsampling. The method we choose to evaluate our proposed approach on is SRCNN, from Dong et al. [3].

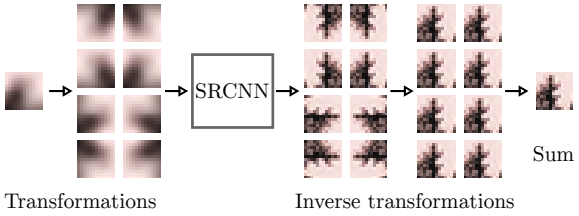

Figure 2. Illustration of the proposed workflow for an algorithm that works on the target resolution (such as SRCNN). The input is transformed, and the eight versions are fed to the network separately. The respective outputs get the corresponding inverted transformation applied, after which they are all averaged.

To exploit the various invariances discussed in this paper, we simply feed the network with all of the reflected and rotated image versions, apply the inverse transformations on the outputs and then average the eight resulting images, as shown in Figure 2, at the cost of additional computations.

3.3. FSRCNN variants

For each pixel in the input image, we need to estimate a subpixel grid of $F \times F$ values (where F is the upscaling factor). As such, a straightforward design of a neural network has it estimating F^2 values for each pixel neighbourhood.

As Figure 3 illustrates, however, we only really need to output a subset of these values - the other values are calculated by feeding the network transformed versions of the input and applying the inverse transformation to the outputs.

For upscaling by factor 2 we only train a network for the top-left sub-pixel. Accounting for 8 possible transformations of the neighbourhood, each of the four sub-pixels is placed in this position exactly twice (see Figure 4). Hence we feed the network these transformed inputs, apply the inverse transformations to the output and then average both estimates for each subpixel location. Upscaling factors 3 and 4 we work the same - the resulting networks both have 3 output values to be trained, as shown in Figures 3 and 4.

3.4. The proposed method and data augmentation

Data augmentation on the training set using the eight aforementioned rigid transformations provides no additional benefit to our proposed method. The improving directions for the weights as inferred from an input image I and its transformed version $H(I)$ are identical, which follows directly from the construction of the network.

4. Results

We evaluate the proposed approach in terms of initial training speed as well as test performance on the Set5, Set14 and DIV2K datasets, both quantitatively and qualitatively. At the time of writing, the DIV2K dataset consists of 800 high-resolution images (with an extra 100 images to be made public after conclusion of the NTIRE17 challenge).

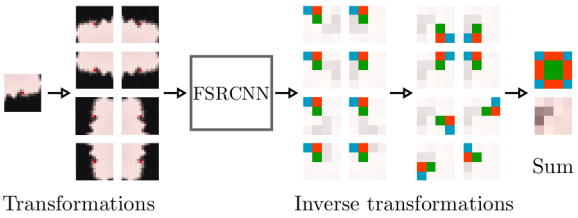

Figure 3. Illustration of the proposed workflow for an algorithm that work on the input resolution (such as FSRCNN) for upscaling factor 4. Again the input is transformed into eight versions, each of which are fed to the network. Where we would typically estimate a 4×4 grid of subpixels, we now only estimate three out of those. All outputs again have their transformation inverted, and after averaging all estimates for the same positions, we see that the entire subpixel grid is covered with only three outputs (down from 16).

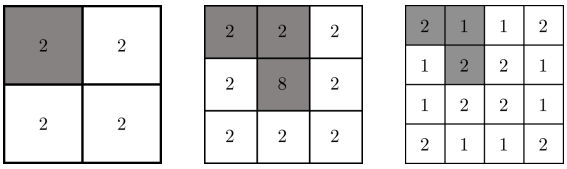

Figure 4. The subpixel grids for upscaling factors 2, 3 and 4 and the corresponding locations estimated by our proposed method (containing the number of estimates per subpixel). See also Figure 3. We only need to estimate the locations above the diagonal in the top left quadrant of the subpixel grid.

Of these 800 images, we use the first 100 as the training set, the next 100 as the testing set and the last 50 as the validation set. The relatively small subset of training images proved to be large enough while still fitting inside the memory of an average workstation.

4.1. Initial training convergence

The networks are trained by randomly extracting 100 patches corresponding with a single input pixel from the images and backpropagating these through the networks. We use the SRCNN network from [3], and the FSRCNN base network with $d = 30$, $s = 10$ and $m = 1$ from [4].

4.1.1 Training SRCNN variants

Here, we evaluate three variants. First of all, we have the exact design from [3], trained only with the images from the dataset. Secondly we have the same network design trained with the images from the dataset augmented by the discussed rigid transformations: a possible reflection and three possible rotations. We call this method SRCNN + data. Finally, we have our proposed variant, SRCNN + invariances, illustrated in Figure 2, which processes all eight transformations of the input image and averages out the reverse transformations of the outputs.

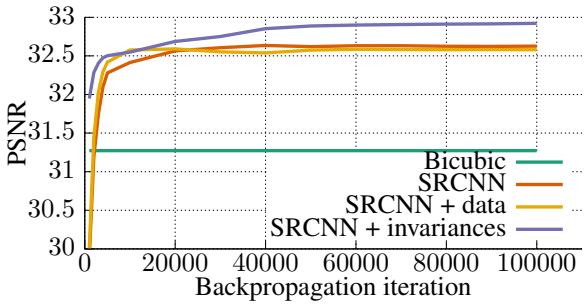

Figure 5. Test PSNR performance in function of the backpropagation iteration for the SRCNN variants (upscaling factor 2).

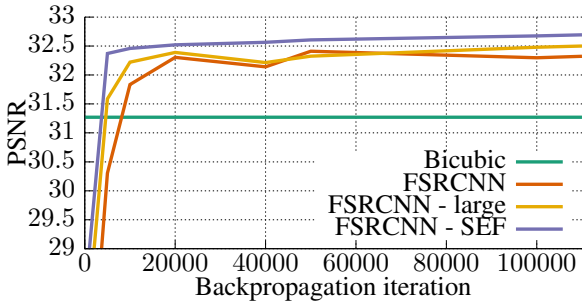

Figure 6. Test PSNR in function of the backpropagation iteration for the FSRCNN variants (upscaling factor 2).

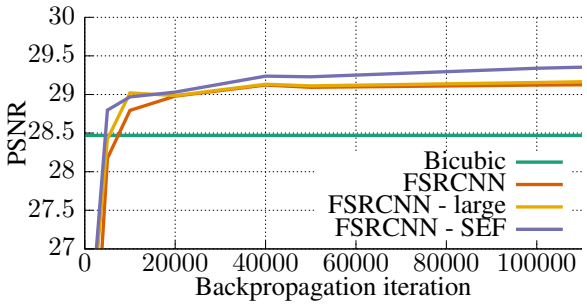

Figure 7. Test PSNR in function of the backpropagation iteration for the FSRCNN variants (upscaling factor 3).

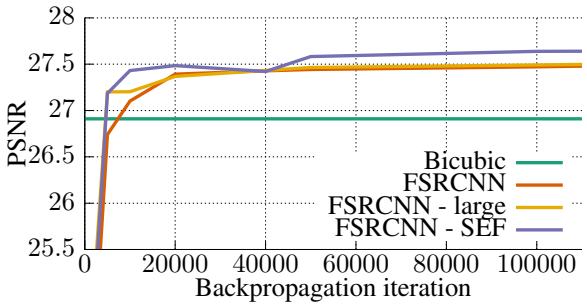

Figure 8. Test PSNR in function of the backpropagation iteration for the FSRCNN variants (upscaling factor 4).

As can be seen in Figure 5, the network variant with internal exploitation of the image invariances converges faster in the initial training iterations. Specifically, it manages to eke out a better performance than the network trained with data augmentation. While data augmentation in the training phase appears not to make a big difference in network performance (indicating that the training set is representative enough), the proposed approach *does* achieve better results.

4.1.2 Training FSRCNN variants

Again, we evaluate three alternative variants. The first variant is the implementation of FSRCNN from [4]. Secondly, we have the proposed variant which only estimates a subset of the subpixel values as shown in Figure 3. This design is called FFSRCN - SEF (Single Entry Filter).

Finally, we also evaluate the FSRCNN network with a correspondingly higher number of hidden nodes. For the factor two upscaling, the FSRCNN - SEF network only needs to estimate a single subpixel value and can leverage all of its internal nodes for this purpose. At the same time, FSRCNN needs to output four values and only has the same number of internal nodes to do so. For this reason, we also evaluate a variant of FSRCNN which has four times the internal nodes: FSRCNN large. This was not meaningful in the previous scenario (working on the upsampled input) as each network only needed to output one per input pixel and the proposed approach gained no significant advantage.

As can be seen in Figure 6, the proposed approach results in faster convergence in terms of iterations. Similar to Dong et al. [4], we also find that an increase of the number of internal nodes does not entail a significant increase in performance. However, there is still a significant difference in performance between FSRCNN large and FSRCNN - SEF, which indicates that the PSNR gain of FSRCNN - SEF compared to FSRCNN is not just due to the relative higher number of internal nodes per output value.

Aside from its better performance compared to SRCNN, the FRSCNN network design is also much more suited for larger upscaling factors. This is because, internally, it works on the input resolution rather than the target resolution (as SRCNN does). As a result, the radii of the convolutions inside the network actually cover a much larger part of the output compared to SRCNN for the same computational cost. Figures 7 and 8 show that the same conclusions can be extended to the higher upscaling factors and that here, too, the proposed method improves upon the original networks.

4.2. Test-time performance of the proposed variants

We now wish to evaluate our proposed approach with converged networks. Gratefully using the pre-trained weights given by Dong et al. for both SRCNN and FSRCNN [3, 4], we again evaluate several variants: from the original pretrained networks to the proposed networks with finetuned weights.

Table 1. Test PSNRs results on Set5 and Set14. See text below for a description of the variants.

Upsampling factor	Dataset	SRF[11]	SCN[15]	SRCNN[3]	SRCNN (us)	SRCNN + I	SRCNN + I + F	FSRCNN-s[4]	FSRCNN-s (us)	FSRCNN-s SEF	FSRCNNs SEF + F	FSRCNN[4]	FSRCNN (us)	FSRCNN SEF	FSRCNN SEF + F
x2	Set5	36.84	36.76	36.33	36.06	36.48	36.65	36.53	36.29	36.35	36.36	36.94	36.69	36.78	36.79
	Set14	32.46	32.48	32.15	31.66	32.04	32.22	32.22	31.74	31.82	31.83	32.54	32.04	32.10	32.10
x3	Set5	32.73	33.04	32.45	32.33	32.69	32.83	32.55	32.38	32.43	32.41	33.06	32.85	32.92	32.92
	Set14	29.21	29.37	29.01	28.78	28.93	29.04	29.08	28.56	28.63	28.62	29.37	28.82	28.86	28.85
x4	Set5	30.35	30.82	30.15	30.26	30.51	30.54	30.04	30.03	30.01	30.08	30.55	30.54	30.57	30.62
	Set14	27.41	27.62	27.21	26.96	27.14	27.20	27.12	26.80	26.82	26.85	27.50	27.08	27.16	27.17

Table 2. Test PSNRs results on Set5 and Set14. See text for a description of the variants.

Upsampling factor	SRCNN (us)	SRCNN + I	SRCNN + I + F	FSRCNN-s	FSRCNN-s SEF	FSRCNNs SEF + F	FSRCNN	FSRCNN SEF	FSRCNN SEF + F
x2	34.61	34.81	34.94	34.75	34.80	34.79	35.06	35.11	35.12
x3	31.41	31.54	31.60	31.41	31.45	31.44	31.66	31.70	31.69
x4	29.61	29.78	29.80	29.64	29.61	29.65	29.88	29.85	29.91

- SRCNN: the original network with its pretrained weights,
- SRCNN + I(nvariances): the average of the reconstructions by the original network with its pretrained weights for all eight discussed transformations,
- SRCNN + I + F(inetuning): finetuning the network weights in the new approach on our DIV2K training set for 100000 backpropagations at stepsize 10^{-6}.
- FSRCNN-s: the original FSRCNN-s network (32,5,1) with its pretrained weights,
- FSRCNN-s + SEF: only using some of the subpixels per prediction location, and using this output as illustrated in Figure 3 to arrive at all subpixel estimates,
- FSRCNN-s + SEF + F: its finetuned version on our DIV2K training set, again for 100000 backpropagations at stepsize 10^{-6}.
- FSRCNN: the original FSRCNN network (56,12,4) with its pretrained weights,
- FSRCNN + SEF: again only using some of the subpixel estimates,
- FSRCNN + SEF + F: its finetuned version, again on our training subset of DIV2K and trained for 100000 backpropagations with stepsize 10^{-6}.

For the original SRCNN, FSRCNN-s and FSRCNN networks we mention both the results from their respective papers and the results obtained within our evaluation framework (which splits up images into patches in order to fit everything on your average GPU). Table 1 shows the performance on Set5 and Set14, listing some other methods for comparison [15, 11]. The conclusion here is that the proposed approach indeed improves slightly on the converged preformance of SRCNN, FSRCNN-s and FSRCNN. For SRCNN further finetuning leads to non-negligible improvements over the pre-trained weights. This is not the case for FSRCNN-s and FSRCNN, where finetuning only leads to negligible changes in the test PSNR values, even though the SEF approach led to slight improvements of the PSNR. We can explain this by remembering that FSRCNN(-s) already explicitly estimates subpixel locations. While our approach for SRCNN + I turns one pixel estimate into 8 estimates and averages those, FSRCNN + SEF turns one subpixel estimate into two estimates and averages those. The same conclusions go for DIV2K, as listed in Table 2. Visual results are available in Figure 9 for the lena and butterfly images from respectively Set14 and Set5. All images except the ones for SRCNN + I + F and FSRCNN SEF + F were extracted from [4].

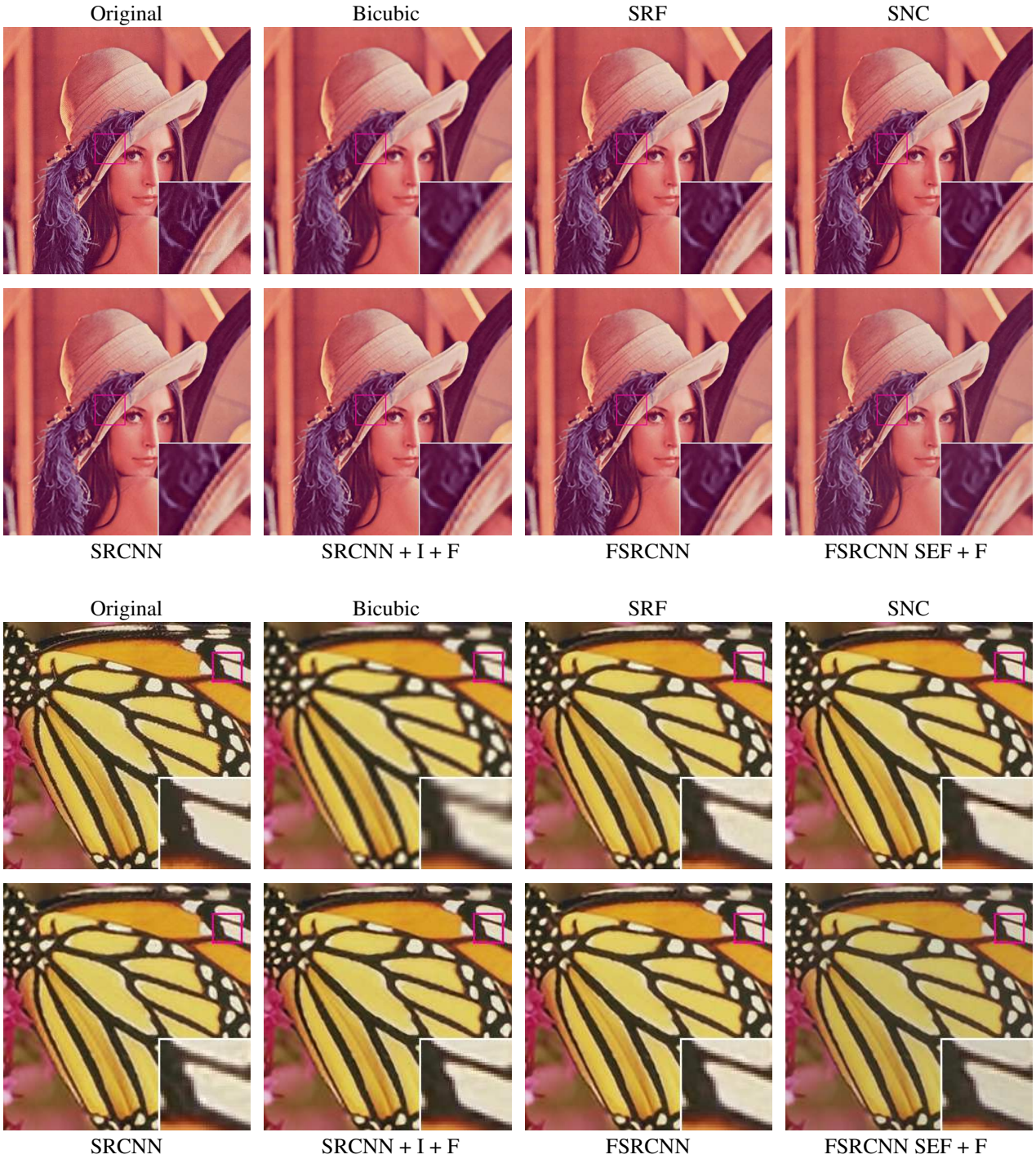

Figure 9. Details of the upsampling result of the butterfly and lena images, for an upsampling factor of 3.

5. Conclusion

We have proposed a new encompassing approach to convolutional neural network superresolution, which can be applied to existing network designs. By exploiting reflectional and rotational invariance of image upsampling, we improve initial training convergence and test-time performance.

We have illustrated this on two distinct approachs to deep learning superresolution: one that takes a rough estimate of the reconstruction as input, and one that starts from the low-resolution image. We have provided qualitative and qualitative evaluations for several datasets. The proposed approach can be easily fit around other existing techniques.

6. Future Work

We see two immediate avenues for possible improvement of the proposed method. First of all, we have restricted ourselves to rigid transformations that respect the sampling grid. Extending the approach to take other possible rotations into account might improve results further, but would also introduce sample theory and the necessity for resampling kernels. These might in their turn introduce new artifacts and issues that hamper performance. And secondly; we simply average all subpixel estimates for the same locations. Better results might be achieved through the use of a more general pooling operator, such as the median [10].

References

[1] A. Adler, Y. Hel-Or, and M. Elad. A shrinkage learning approach for single image super-resolution with overcomplete representations. *Computer Vision–ECCV 2010*, pages 622–635, 2010. 2

[2] S. Dai, M. Han, W. Xu, Y. Wu, Y. Gong, and A. K. Katsaggelos. Softcuts: a soft edge smoothness prior for color image super-resolution. *IEEE Transactions on Image Processing*, 18(5):969–981, 2009. 2

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014. 1, 2, 3, 4, 5

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb 2016. 1, 2, 3, 4, 5

[5] R. Fattal. Image upsampling via imposed edge statistics. In *ACM Transactions on Graphics (TOG)*, volume 26, page 95. ACM, 2007. 2

[6] P. P. Gajjar and M. V. Joshi. New learning based super-resolution: Use of dwt and igmrf prior. *IEEE Transactions on Image Processing*, 19(5):1201–1213, 2010. 2

[7] W. Gong, Y. Tang, X. Chen, Q. Yi, and W. Li. Combining edge difference with nonlocal self-similarity constraints for single image super-resolution. *Neurocomputing*, pages –, 2017. 2

[8] Y. Huang and Y. Long. Super-resolution using neural networks based on the optimal recovery theory. In *2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 465–470, Sept 2006. 2

[9] X. Ji, Y. Lu, and L. Guo. Image super-resolution with deep convolutional neural network. In *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, pages 626–630, June 2016. 2

[10] H. Q. Luong, A. Ledda, and W. Philips. Non-local image interpolation. In *2006 International Conference on Image Processing*, pages 693–696, Oct 2006. 7

[11] S. Schulter, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3799, 2015. 5

[12] W.-Z. Shao and Z.-H. Wei. Edge-and-corner preserving regularization for image interpolation and reconstruction. *Image and Vision Computing*, 26(12):1591–1606, 2008. 2

[13] R. Timofte, R. Rothe, and L. Van Gool. Seven ways to improve example-based single image super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1865–1873, 2016. 2

[14] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 370–378, 2015. 2

[15] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 370–378, 2015. 2, 5

[16] M.-C. Yang, D.-A. Huang, C.-Y. Tsai, and Y.-C. F. Wang. Self-learning of edge-preserving single image super-resolution via contourlet transform. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on*, pages 574–579. IEEE, 2012. 2