Proc. 21st Int. Conf. on Neural Information Processing (ICONIP 2014), pp. 250–257.

Image Super-Resolution with Fast Approximate Convolutional Sparse Coding

Christian Osendorfer*, Hubert Soyer*, and Patrick van der Smagt

Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik und Echtzeitsysteme, Boltzmannstraße 3, 85748 München

Abstract. We present a computationally efficient architecture for image super-resolution that achieves state-of-the-art results on images with large spatial extend. Apart from utilizing Convolutional Neural Networks, our approach leverages recent advances in fast approximate inference for sparse coding. We empirically show that upsampling methods work much better on latent representations than in the original spatial domain. Our experiments indicate that the proposed architecture can serve as a basis for additional future improvements in image super-resolution.

Keywords: Image Processing, Sparse Coding, Convolutional Neural Networks

1 Introduction

The term super-resolution in computer vision generally denotes the process of increasing the resolution of a given image or a set of images. Sparse Coding, a powerful dictionary learning method [1–4], was recently applied to single-image super-resolution in a very successful way [5–8]. Hereby, the sparse code couples two different kinds of dictionaries: One dictionary contains low-resolution atoms and one dictionary contains high-resolution atoms. Super-resolving an image patch is then performed in a straight-forward manner: Given the low-resolution patch, determine its sparse code relative to the low-resolution dictionary, and then apply this sparse code in the high-resolution generative model. Couzinie-Devy et al. [5] apply this idea for deblurring and super-resolution by processing each input image patch by patch. Yang et al. [6] follow a similar idea but propose different additions for face and natural images and combine their method with a global reconstruction constraint over the whole image. [7] and [8] take sparse coding for super-resolution even further, working not only with two dimensional images but processing even depth information.

Note that these approaches resolve the super-resolution problem in a very elegant *implicit* way: Upsampling the spatial data, as is necessary in the standard super-resolution approaches like bicubic interpolation [9], is achieved indirectly with the high-resolution dictionary. However, this entails that applying these methods to images larger than the training patches is cumbersome and computationally inefficient. Furthermore, finding the sparse code for a given image

patch is a costly optimization problem itself and thus applying the mentioned approaches to large images with many patches is extremely slow. Yang et al. specify the time to enlarge a 85×86 image to 255×258 with their sparse coding model from [6] and a reasonably chosen set of parameters as approximately 30 seconds on a Core duo@1.83 Ghz with 2GB Ram.

We tackle exactly this problem: Our proposed super-resolution architecture leverages recent insights into fast approximate sparse coding and utilizes the natural characteristic of the convolutional operator. In this way, we can *train our model on exemplary image patches and scale it to arbitrarily sized test images without any additional cost*. We present our approach and the necessary preliminary work in section 2. Experimental details and results are described in section 3. Section 4 concludes with a brief outlook on future work.

2 Approach

Recently, Convolutional Neural Networks (CNN) [10] have gained a lot of attention due to their success in several large scale computer vision tasks [11–13]. Due to the nature of the convolutional operator, CNNs can be applied to inputs of arbitrary size, i.e. they are apriori not tied to the dimensionality of the samples from the training set. This property is often overlooked (see [14, 15] for some notable recent exceptions), yet is one of the main ingredients in order to allow the transfer of learned *patch-based super-resolution* to *full image super-resolution*. However, the standard approach of the previously mentioned sparse coding based super-resolution methods is now no longer applicable: The *upscaling* of the data is encoded in the dictionary elements of these methods – this is not possible in a straight forward manner with a convolutional based approach.

Where could upsampling of an image happen? The common approach [9] is to upsample in the image domain. The problem of super-resolution then simply reduces to *learning an optimal deconvolutional* operator. However, if one considers the latent representation of an image (i.e. the convolutional sparse codes in our case), another option occurs: Upsampling this latent representation. Similar to standard signal processing we hypothesize that upsampling should be performed on the *adaptively learned* latent representation of an image and not on its original spatial representation.

Specifically our method consists of three parts: (i) Fast convolutional sparse coding of an input, (ii) upsampling of the sparse codes and (iii) convolutional decoding of the upsampled sparse codes. If the upsampling method is chosen in the right way, this architecture can be applied to inputs (i.e. images) of arbitrary dimensions.

2.1 Fundamentals

In recent years, a wide variety of sparse coding algorithms were developed that learn good feature representations of natural images [1–4]. A big practical hindrance of the standard sparse coding algorithms is that inference of a sparse code

requires running a computationally expensive optimization algorithm. Utilizing the powerful approximation capabilities of neural networks, [16, 17] propose an algorithm that can simultaneously learn an overcomplete dictionary for sparse coding and an approximator that predicts the optimal sparse representation.

Taking this idea even further [18] shows that by introducing convolutional operators a richer, more diverse set of features is learned. The objective function for their architecture is as follows:

$$\mathcal{L}(x, z, \mathcal{D}^{(d)}, \mathcal{D}^{(e)}) = \frac{1}{2} \underbrace{\|x - \sum_{k=1}^{K} \mathcal{D}_k^{(d)} * z_k\|_2^2}_{\text{decoder}} + \tag{1}$$

$$\underbrace{\sum_{k=1}^{K} \|z_k{}^\star - f(\mathcal{D}_k^{(e)} * x)\|_2^2}_{\text{encoder}} + \underbrace{\lambda |z|_1}_{\text{sparsity}}$$

where $x \in \mathbb{R}^{m \times n}$ is an input image, $z \in \mathbb{R}^{K \times o \times p}$ is a set of K many (2d) sparse codes (of dimension $o \times p$ each) with $z_k{}^\star$ being a version of sparse code k that is optimal with respect to the decoder part of the loss function. $\mathcal{D}^{(e)}$ and $\mathcal{D}^{(d)}$ are sets of encoder/decoder filters, f is a non-linear function, $*$ denotes convolution and $|z|_1$ is the l_1 norm over all sparse codes z. The *decoder* part combined with $|z|_1$ is equivalent to the standard convolutional sparse coding formulation. The *encoder* tries to produce representations that are similar to the optimal convolutional sparse codes. Given an input image, the encoder produces its corresponding sparse code and can therefore be seen as a single step approximator of the iterative sparse code optimization method, outperforming it significantly in speed.

Learning (i.e. finding $\mathcal{D}^{(e)}$ and $\mathcal{D}^{(d)}$) happens in an alternating manner: (i) First, by keeping $\mathcal{D}^{(e)}$ and $\mathcal{D}^{(d)}$ constant, minimize eq. 1 with respect to z. Starting from the initial value provided by $f(\mathcal{D}_k^{(e)} * x)$ (for all k) this can be done with various kinds of optimization algorithms. In our experiments, we employed the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [19]. (ii) Second, based on this optimal sparse code, update $\mathcal{D}^{(e)}$ and $\mathcal{D}^{(d)}$ by one step of gradient descent.

2.2 Upsampling

Convolution produces results similar in size to its input when applied to an image. Naïvely employing the architecture from eq. 1 for super-resolution can therefore only be managed with a trick: Given a low-resolution image patch the encoder approximates a sparse code. The decoder then uses this sparse code to infer a high resolution of the *patch center only*. Super-resolving an image is then achieved by applying this process repeatedly to different areas of the low-resolution image followed by stacking together the results.

However, in order to have enough information in the sparse representation for upsampling the patch center, the filter sizes in the *encoder* have to be choosen

very large in relation to the low-resolution image, which usually leads to learning averaging filters only. As expected, this idea yields very poor results which resemble only a very blurred upscaled version of the low-resolution image center.

For proper image super-resolution the model from eq. 1 requires some modifications: As already mentioned before hand, we introduce an upscaling layer between the encoding and decoding stage of the model, working on the sparse representation of an input:

$$\mathcal{L}(x^{(lr)}, x^{(hr)}, z, \mathcal{D}^{(d)}, \mathcal{D}^{(e)}, W) = \tag{2}$$
$$\underbrace{\frac{1}{2}\|x^{(hr)} - \sum_{k=1}^{K} \mathcal{D}_k^{(d)} * \widehat{W\overline{z}_k}\|_2^2 +}_{\text{decoder}}$$
$$\underbrace{\sum_{k=1}^{K} \|z_k{}^{\star} - f(\mathcal{D}_k^{(e)} * x^{(lr)})\|_2^2 +}_{\text{encoder}} \underbrace{\lambda|z|_1}_{\text{sparsity}}$$

where $x^{(lr)}$ and $x^{(hr)}$ are the low-resolution and high-resolution versions of the input x respectively and $W \in \mathbb{R}^{o_{(hr)} \cdot p_{(hr)} \times o_{(lr)} \cdot p_{(lr)}}$ is a matrix that scales the *flattened* sparse codes $\overline{z}_k$ up to their high-resolution version. After applying W the result has to be reshaped to the correct high-resolution sparse code shape $o_{(hr)} \times p_{(hr)}$ as denoted by $\widehat{W\overline{z}_k}$. This formulation of the model allows to use any upsampling method that is based on a linear transformation by choosing matrix W accordingly. Note that learning $\mathcal{D}^{(e)}$ and $\mathcal{D}^{(d)}$ proceeds exactly as in the original architecture from eq. 1. Figure 1 shows a graphical interpretation of the model with input data at different stages of the pipeline.

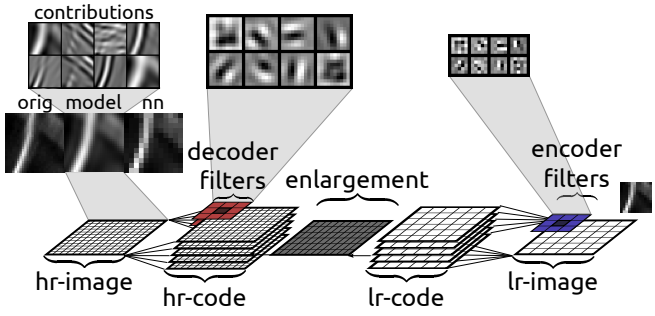

Fig. 1: On the right side, a low-resolution image is feed into the fast approximate convolutional sparse coding module, then upscaled and finally deconvoluted. See section 3.2 for more comments with respect to this Figure.

Albeit an arbitrarily structured matrix W would be the most flexible and powerful approach, the upsampling matrix W must be choosen as a convolu-

tional operator itself. In our experiments, we considered four different kinds of upscaling operations for the sparse codes: bilinear interpolation, linear shifted interpolation, nearest neighbor interpolation and our own, non-standard, perforate interpolation. Figure 2 illustrates these methods graphically for the example of two-fold super-resolution: It shows the convolutional weights that are applied to a neighborhood of pixels in a low-resolution sparse code in order to generate a pixel in the high-resolution sparse code.

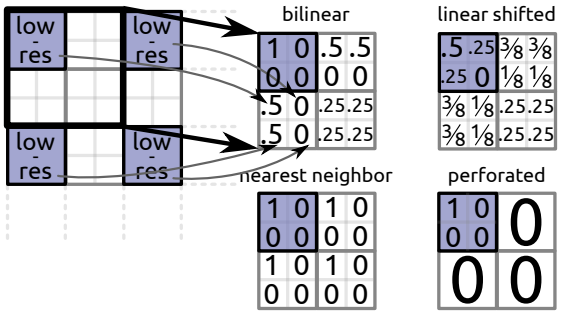

Fig. 2: Upsampling an image by a factor of 2: Every pixel in the low-resolution image is replaced by 4 pixels, indicated by the black square in the left-most image. How the values of these 4 pixels are actually computed depends on the specific upsampling scheme. Here we consider schemes that utilize 4 neighboring low-resolution pixels to compute one high-resolution pixel. To the right, we show the upsampling weights for the 4 methods mentioned in the text. Note that *perforated* upsampling induces additionally sparsification. It crudely approximates the *inverse of the widely-used max-pooling operator* from deep CNNs [12].

3 Experiments

Most super-resolution approaches rely on datasets with very low resolutions [6, 20, 21]. However, the strength of the presented model lies in its speed and applicability to large images. Thus, we chose to train and evaluate on a dataset that features images with very high resolutions, the Van Hateren dataset [22]. It comprises 4167 gray scale images with 1536×1024 pixels each and a gray scale depth of 12 bit. The pictures mostly depict scenes from nature or buildings. For the training set we extracted 20 patches of size 50×50 at random positions from 400 images, resulting in 8000 training samples. In order to generate the low-resolution patches, the original ones were blurred with an anti-aliasing filter and then down-sampled accordingly. The validation set was created in the same way but using a different set of 100 images. And finally the test set features 100 *unseen full-sized* images. Training and applying the other sparse coding based super-resolution algorithms cited earlier would not be tractable on (test)

images of this size. We therefore compared our approach with standard super-resolution algorithms from the image processing domain [9]: bicubic spline based interpolation, bilinear interpolation and nearest neighbor interpolation. After training the model from eq. 2 is finished, we further *fine-tuned* the complete convolutional super-resolution model.

3.1 Training details

To keep training time manageable the model was trained one sample (that is, a pair of low resolution and accompanying high resolution image patch) at a time, samples were chosen at random. The sparse codes were optimized with 5 iterations of FISTA at each training step. Less than 5 FISTA iterations decreased the final results noticeably while more iterations didn't have any influence on the results but increased training time significantly. The non-linearity f (see eq. 1) is set to a *soft threshold function* [17]. Filters were optimized with one step of gradient descent per model training iteration. All experiments were trained for 1 million epochs with an initial learning rate of $2 \cdot 10^{-4}$ that decayed as $\frac{2 \cdot 10^{-4}}{1.0 + (epoch/5000)}$. Finally, the model with the lowest objective function score on the validation set was further fine tuned with a learning rate of $1 \cdot 10^{-6}$ for another 16000 epochs.

3.2 Evaluation

There are a number of ways to evaluate the results of super-resolution: Some papers judge the quality of the results by their Mean Squared Error per pixel (MSE) to the ground truth [6], some use the related Peak-Signal-to-Noise Ratio (PSNR) [23] and others rely on Structured Similarity (SSIM) as a measure of error [20, 21]. PSNR is logarithmically proportional to MSE and both can be argued to only inaccurately represent the human understanding of better or worse regarding the quality of an image-reconstruction. SSIM aims to tackle this shortcoming – we therefore report both PSNR and SSIM scores in our evaluation.

A qualitative impression of the learned architecture is shown in Figure 1: Typical filters for both the encoder as well as the decoder are shown, in this case for an architecture with 8 latent channels. On the left side, a super-resolved image patch (denoted by *model*) is shown, computed from a low-resolution image patch depicted at the right side. For comparison, the original (*orig*) high-resolution patch and the nearest neighbor interpolation (*nn*) is also shown.

Table 1 shows both the PSNR and SSIM scores for the various types of latent upsampling methods presented in section 2.2 (CNN-PF denotes *perforated interpolation*, CNN-BL denotes *bilinear interpolation*, CNN-NN denotes *nearest neighbor interpolation* and CNN-SH denotes *linear shifted interpolation*). BCI, BLI and NNI denote the classic bicubic, bilinear and nearest neighbor interpolation methods in the original spatial domain respectively. K, the number latent channels is set to 8 in all experiments: Smaller numbers (4, 6) resulted in inferior results, for larger numbers (12, 16, 32) training did not converge after 21

	CNN-PF	CNN-BL	CNN-NN	CNN-SH	CNN-LD	BCI	BLI	NNI
PSNR	**32.55**	32.52	32.49	31.98	32.07	31.80	30.79	30.55
SSIM	**0.946**	0.945	0.942	0.941	0.944	0.935	0.922	0.913

Table 1: Our perforated sparse code upsampling method performs best. For a full image from the test set the unoptimized version takes about 4 seconds, compared to approximately 2.5 seconds for bicubic interpolation. Larger numbers are better for both PSNR and SSIM.

days and thus was stopped – FISTA proved to be the bottleneck for these larger models. All other hyperparameters were determined via the validation set. We also learned W (see eq. 2), which is resembled by the column CNN-LD.

Apart from the fact that CNN-PF outperforms all other approaches, in particular the widely used bicubic interpolation method, we point out the following two observations: (i) Both bilinear and nearest neighbor interpolation methods perform significantly better when applied to the latent representation, supporting our original hypothesis empirically. Hence, an obvious next step is to apply bicubic interpolation accordingly in the latent domain – however this can't no longer be written in the form of eq. 2 because now *non-linear features* need to be computed in the sparse domain. (ii) Fine-tuning did not improve the results. We assume that this is due to approximating FISTA with only one convolutional layer.

4 Summary and Outlook

We presented a single image super-resolution approach based on fast approximate sparse coding with convolutional neural networks. Our approach not only outperforms state-of-the-art super-resolution methods for large images but is also computationally efficient. As indicated by our experiments, unrolling the iterative convolutional FISTA algorithm in a way similar to [24] is a very promising future research direction. Extrapolating the observations from Table 1, latent bicubic upsampling, or even more general upsampling methods that can be realized through [25] should increase the performance of our framework considerably.

References

1. Olshausen, B.J., Field, D.J.: Sparse coding with an over complete basis set: a strategy employed by v1? Vision Research (1997)
2. Mairal, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A.: Discriminative learned dictionaries for local image analysis. In: CVPR. (2008)
3. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. (2007)

4. Ranzato, M.A., Poultney, C., Chopra, S., Lecun, Y.: Efficient learning of sparse representations with an energy-based model. In: NIPS. (2006)
5. Couzinie-Devy, F., Mairal, J., Bach, F., Ponce, J.: Dictionary learning for deblurring and digital zoom. arXiv preprint arXiv:1110.0957 (2011)
6. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. Image Processing, IEEE Transactions on **19**(11) (2010)
7. Hu, T., Nunez-Iglesias, J., Vitaladevuni, S., Scheffer, L., Xu, S., Bolorizadeh, M., Hess, H., Fetter, R., Chklovskii, D.: Super-resolution using sparse representations over learned dictionaries: Reconstruction of brain structure using electron microscopy. arXiv preprint arXiv:1210.0564 (2012)
8. Zhang, Y., Wu, G., Yap, P.T., Feng, Q., Lian, J., Chen, W., Shen, D.: Reconstruction of super-resolution lung 4d-ct using patch-based sparse representation. In: CVPR. (2012)
9. Petrou, M., Petrou, C.: Image Processing: The Fundamentals. Wiley & Sons (2010)
10. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE **86** (1998)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
12. Ciresan, D.C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: CVPR. (2012)
13. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)
14. Masci, J., Giusti, A., Ciresan, D.C., Fricout, G., Schmidhuber, J.: A fast learning algorithm for image segmentation with max-pooling convolutional networks. In: ICIP. (2013)
15. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: ICLR. (2014)
16. Kavukcuoglu, K., Ranzato, M., LeCun, Y.: Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU (2008)
17. Kavukcuoglu, K., Ranzato, M., Fergus, R., Le-Cun, Y.: Learning invariant features through topographic filter maps. In: CVPR. (2009)
18. Kavukcuoglu, K., Sermanet, P., Boureau, Y.L., Gregor, K., Mathieu, M., Cun, Y.L.: Learning convolutional feature hierarchies for visual recognition. In: NIPS. (2010)
19. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences **2**(1) (2009) 183–202
20. He, L., Qi, H., Zaretzki, R.: Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution. In: CVPR. (2013)
21. Lu, X., Yuan, H., Yan, P., Yuan, Y., Li, X.: Geometry constrained sparse coding for single image super-resolution. In: (CVPR. (2012)
22. Hateren, J.H.v., Schaaf, A.v.d.: Independent component filters of natural images compared with simple cells in primary visual cortex. Proceedings: Biological Sciences **265**(1394) (1998)
23. Zhang, K., Gao, X., Tao, D., Li, X.: Multi-scale dictionary for single image super-resolution. In: CVPR. (2012)
24. Sprechmann, D., Bronstein, A.M., Sapiro, G.: Learning efficient sparse and low rank models. arXiv preprint arXiv:1212.3631 (2012)
25. Lin, M., Chen, Q., Yan, S.: Network in network. In: ICLR. (2014)