A Novel Approach for Image Super Resolution Using Kernel Methods

Adhish Prason^{1(✉)}, Himanshu Chaubey¹, Abhinav Gupta¹,
Rohit Garg¹, and Santanu Chaudhury²

¹ Samsung Research and Development Institute Delhi, Noida, India
adhishprason@gmail.com

² Department of Electrical Engineering, Indian Institute of Technology Delhi,
Delhi, India

Abstract. We present a learning based method for image super resolution problem. Our approach uses kernel methods to build an efficient representation and also to learn the regression model. For constructing an efficient set of features, we apply Kernel Principal Component Analysis (Kernel-PCA) with a Gaussian kernel on a patch based data-base constructed from 69 training images up-scaled using bi-cubic interpolation. These features were given as input to a non-linear Support Vector Regression (SVR) model, with Gaussian kernel, to predict the pixels of the high resolution image. The model selection for SVR was performed using grid search. We tested our algorithm on an unseen data-set of 13 images. Our method out-performed a state-of the-art method and achieved an average of 0.92 dB higher Peak signal-to-noise ratio (PSNR). The average improvement in PSNR over bi-cubic interpolation was found to be 3.38 dB.

1 Introduction

The aim of the super resolution methods is to increase the resolution of low resolution (LR) images. The motivations behind increasing interest in super resolution methods are their potential to improve the resolution of the images taken by the low cost imaging devices and also to exploit the full capability of the high resolution (HR) displays. Recently machine learning techniques have been increasingly used to solve the problem of image super-resolution. Kernel methods [1] have been successfully employed for various categorization tasks in image processing domain. In this paper we present a novel approach to solve the image super resolution problem using two such methods; kernel principal component analysis (kernel-PCA) and support vector regression (SVR). Below, we discuss some other important contributions which use machine learning techniques for solving the image super resolution problem. Freeman et al. [2] use Markov Random Fields and Belief Propagation to predict the high resolution image corresponding to a low resolution image. Chang et al. [3] propose a method inspired by manifold learning using Locally Linear Embedding (LLE). Their method assumes that the similar manifolds are formed in the low resolution and

the high-resolution patch space. Yang et al. [4] proposes a sparse representation based method. Their method is inspired by the fact that a sparse linear combination of elements from over-complete dictionary can represent image patches. They find a sparse representation for the patches of the low resolution image and use this representation to obtain corresponding high resolution image. Due to the excellent results and a robust technique, Yang et al.'s method [4] is certainly one of the state-of-the-art methods. We compare our results to their method. The reason we chose their method to be the benchmark for our study is that apart from being one of the state-of-the-art methods, they have provided their code [5], the training data and their pre-learned dictionaries which can be directly used to have fair comparison between the two methods. Apart from the above methods, SVR has also been used for the purpose of super-resolution. Ni et al. [6] apply SVR in the DCT domain. An et al. [7] have also used SVR for image super-resolution. However, our method is different from their method [7] as our feature set is entirely different. We have applied kernel-PCA for feature extraction while [7] utilizes simply the raw pixels along with center pixel's gradient with weights as their features. Ni et al. [6] also apply SVR for image super resolution but the framework and motivation of their usage of SVR is entirely different from our method. Yuan et al. [8] use sparse solution of Kernel-PCA applied on HR image patches to determine the coefficients which can represent higher-order statistics of HR image structures. Further Yuan et al. [8] constructs dictionaries and maps an LR image patch to the coefficients of the respective HR image patch. Kernel-PCA is also a part of our method. However, we have applied it on LR image patches in order to extract a powerful set of features which are eventually fed to yet another kernel based method (Support Vector Regression).

The next section explains our method in detail. We start the next section with a brief introduction to kernel-PCA and support vector regression followed by the detailed explanation of our super resolution method. In Sect. 3, we discuss our experiments and present the results. We conclude the paper in Sect. 4.

2 Method

Kernel trick [1] is one of the most powerful tools in machine learning and has been the backbone of many algorithms. In this work we combine two such algorithms, Kernel Principal Component Analysis [9] (kernel PCA) and Support Vector Regression (SVR) to solve the problem of single-image super-resolution. In the next two subsections we give a brief introduction to kernel PCA and Support Vector Regression (SVR) [10, 11] respectively. Subsequently, we will explain our method which uses kernel-PCA and SVR for single image super-resolution.

2.1 Kernel PCA

Before moving on to kernel PCA let us have a brief overview of PCA. PCA is a linear method which is often used for feature extraction. The idea behind it

is to transform a dataset having correlated features in such a way that the new features become linearly uncorrelated. Let $\mathbf{P}$ be an $d \times n$ sized matrix, where each column represents an example with d features and n be the number of training examples. Let $\mathbf{p}^{(i)}$ be the i th example. Here we assume that data-set has zero mean across all the examples, for each feature. Let $\mathbf{Q}$ be the covariance matrix given as

$$\mathbf{Q} = \frac{1}{n} \sum_{i=1}^n \mathbf{p}^{(i)} (\mathbf{p}^{(i)})' \quad (1)$$

where $(\mathbf{p}^{(i)})'$ is the transpose of vector $\mathbf{p}^{(i)}$. Let $\mathbf{E}$ be the $d \times d$ sized matrix, where j th column represents j th eigen-vector $\mathbf{e}^{(j)}$ of $\mathbf{Q}$. The new transformed data matrix with uncorrelated features is given as

$$\mathbf{P}_N = \mathbf{E}' * \mathbf{P} \quad (2)$$

Although Principal Component Analysis (PCA) is an appropriate method for dimensionality reduction and feature extraction if the data is linearly-separable, in many cases, where the data is not linearly separable, we need a non-linear feature extraction method. So, if we apply a non-linear mapping to the input features, the data have higher chance to be linearly separable in the new feature space. Let $\mathcal{F}$ be the non-linear mapping, which transforms the input $\mathbf{p}^{(i)}$ to a very high-dimensional space. The new covariance matrix can be given as

$$\mathbf{Q}_N = \frac{1}{n} \sum_{i=1}^n \mathcal{F}(\mathbf{p}^{(i)}) (\mathcal{F}(\mathbf{p}^{(i)}))' \quad (3)$$

For transforming the non-linearly mapped data the same way as PCA, we need to calculate the eigen-vectors of the covariance matrix $\mathbf{Q}_N$. Unfortunately, calculating the eigen-vectors and eigen-values of such a huge matrix is computationally not feasible. However, it has been shown [9] that the projections of the data points onto the eigen-vectors in the new high dimensional space can be obtained without even calculating the high dimensional features $\mathcal{F}(\mathbf{p}^{(i)})$; using the *kernel-trick* [1]. Let $\mathbf{e}_N^{(j)}$ be the j th eigen-vector of the covariance matrix $\mathbf{Q}_N$. Let $\mathbf{p}^{(t)}$ be a new data-point and $\mathcal{F}(\mathbf{p}^{(t)})$ be the corresponding data-point in the high-dimensional space. As proved in [9], projection of $\mathcal{F}(\mathbf{p}^{(t)})$ onto $\mathbf{e}_N^{(j)}$, can be written as

$$\langle \mathbf{e}_N^{(j)}, \mathcal{F}(\mathbf{p}^{(t)}) \rangle = \sum_{i=1}^n v(i)^{(j)} \langle \mathcal{F}(\mathbf{p}^{(t)}), \mathcal{F}(\mathbf{p}^{(i)}) \rangle \quad (4)$$

where $v(i)^{(j)}$ is the i th element of the column vector $\mathbf{v}^{(j)}$, and $\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(j)}, \dots, \mathbf{v}^{(n)}$ are the solutions of the equation

$$\beta \mathbf{v} = \mathbf{K}_\kappa \mathbf{v} \quad (5)$$

$\mathbf{K}_\kappa$ being an $n \times n$ sized matrix whose (i, j) th element is given by $\langle \mathcal{F}(\mathbf{p}^{(i)}) \mathcal{F}(\mathbf{p}^{(j)}) \rangle$. The kernel trick says [1] that the dot product in the high dimensional space (obtained using a non-linear mapping) can be calculated using a kernel function. A kernel function is a dual input function which follows the Mercer's condition. Further, the methods which use kernel function and kernel-trick are classified as kernel methods in machine learning. Let k_κ be the kernel function, given as $k_\kappa(x, y) = \langle \mathcal{F}(\mathbf{x}) \mathcal{F}(\mathbf{y}) \rangle$. It can be seen that the two Eqs. (4) and (5) are written in terms of the dot products in the non-linear feature space, which eventually are nothing but the kernel function evaluations for the input values. Replacing, the dot product with the kernel function in Eq. (4), we have

$$\langle \mathbf{e}_N^{(j)}, \mathcal{F}(\mathbf{p}^{(t)}) \rangle = \sum_{i=1}^n v(i)^{(j)} k_\kappa(\mathcal{F}(\mathbf{p}^{(t)}), \mathcal{F}(\mathbf{p}^{(i)})) \quad (6)$$

Further, the matrix $\mathbf{K}_\kappa$ has to be modified in such a way that the data in the high dimensional space becomes approximately centered and the eigen-vectors of the covariance matrix $\mathbf{Q}_N$ are normalized, [9, 12]. In the next section we give a brief overview of another popular kernel based method, i.e. support vector regression.

2.2 Support Vector Regression

Support Vector Regression [10, 13] was developed as an extension of Support Vector Machines [11], which are one of the most powerful algorithms in machine learning and have excellent generalization ability.

Let $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$ be the training dataset of n examples with $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ being n input variables and $y_1, y_2, \dots, y_n$ being corresponding target variables. The dual optimization function for linear SVR training can be written as

$$\left. \begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} & \left[\sum_{i=1}^n (\alpha_i - \alpha_i^*) (y_i) - \frac{1}{2} \sum_{i,j=1}^n (\alpha_i - \alpha_i^*) (\alpha_j - \alpha_j^*) \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^n (\alpha_i + \alpha_i^*) (\epsilon_i) \right] \\ \text{subject to } & \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad C \geq \alpha_i, \alpha_i^* \geq 0, \quad i = 1, \dots, n \end{aligned} \right\} \quad (7)$$

where $\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$ are the dual variables whose i th elements are given as α_i, α_i^* ; C is the regularization parameter and ϵ represents the maximum permissible training error. Now to convert linear SVR to a non-linear SVR, let ϕ be the nonlinear mapping applied to the inputs. The new optimization problem can be written as

$$\left. \begin{aligned} \max_{\alpha, \alpha^*} & \left[\sum_{i=1}^n (\alpha_i - \alpha_i^*) (y_i) - \frac{1}{2} \sum_{i,j=1}^n (\alpha_i - \alpha_i^*) (\alpha_j - \alpha_j^*) \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle - \right. \\ & \left. \sum_{i=1}^n (\alpha_i + \alpha_i^*) (\epsilon_i) \right] \\ \text{subject to } & \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad C \geq \alpha_i, \alpha_i^* \geq 0, \quad i = 1, \dots, n \end{aligned} \right\} \quad (8)$$

Now, as the above equation is written in terms of dot products in the high dimensional feature space, the kernel trick is exploited to solve the dual optimization problem without even calculating the new features in the high dimensional space. Let k_s be the kernel function defined as $k_s(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$. Thus finally inserting the kernel function in place of $\langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$, the above equation can be written as

$$\left. \begin{aligned} \max_{\alpha, \alpha^*} & \left[\sum_{i=1}^n (\alpha_i - \alpha_i^*) (y_i) - \frac{1}{2} \sum_{i,j=1}^n (\alpha_i - \alpha_i^*) (\alpha_j - \alpha_j^*) k_s(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) - \right. \\ & \left. \sum_{i=1}^n (\alpha_i + \alpha_i^*) (\epsilon_i) \right] \\ \text{subject to } & \sum_{i=1}^n (\alpha_i - \alpha_i^*) = 0, \quad C \geq \alpha_i, \alpha_i^* \geq 0, \quad i = 1, \dots, n \end{aligned} \right\} \quad (9)$$

The solution of the dual optimization function is finally used for predicting the output value of a new test pattern. In the following, we discuss our approach which applies the two kernel methods discussed above (kernel PCA and SVR), for image super-resolution.

2.3 Application of Kernel PCA and SVR to Single Image Super-Resolution

For any regression task to perform well we need features with high discrimination ability and a strong regression method. In our method for image super resolution we use kernel methods, i.e. kernel PCA for extracting a strong feature set and SVR for predicting the intensity values of the high resolution (HR) image. Figure 1 depicts our method in detail.

Feature Extraction. Due to high sensitivity of human vision system to luminance changes in comparison to the color changes, we converted the images from *RGB* color space to *YCbCr* color space. We applied our method on the *Y* component only, while *Cb* and *Cr* components are directly taken from up-scaled images using bi-cubic interpolation. For extracting an efficient set of features, we determine a mapping obtained by applying kernel PCA on a patch based

Fig. 1. Block diagram for the proposed method

training data-set constructed using 69 low resolution images. Each of the low resolution images was first up-scaled by a scaling factor of 2 using bi-cubic interpolation. We extracted patches of size 3×3 , centered around the locations randomly selected from the up-scaled image. The Y values at the same locations in the high resolution images were also saved to be used as the target values for training our regression model. Moreover, patches were also extracted from the same locations in gradient image of up-scaled image (using Sobel's operator). Thus for each randomly selected location, we extracted a patch pair. Both the patches of each patch-pair were vectorized and concatenated to get our 18 dimensional input examples. Overall, from 69 images, we extracted 90000 examples which were used to learn our regression models. Earlier, we have seen that kernel PCA needs to perform the eigen-value analysis on the gram matrix $\mathbf{K}_\kappa$, which is of size $n \times n$, n being the total number of examples. Thus, in order to reduce the memory and computational complexity, we performed kernel PCA on a much smaller data-set of 1900 examples to obtain the solutions of the Eq. (5). Finally, these solutions were used to extract features from 90000 examples (as in Eq. (6)). The kernel we used for feature extraction was Gaussian kernel, $k_\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2})$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{18}$. The standard deviation σ was chosen to be the mean of the pair-wise distance of all the possible example pairs. Total 18 features were extracted to construct our training data-set, which was eventually given as input to learn the SVR model.

Fig. 2. Comparison of results obtained for the “Leaves” image. Left image in the upper row shows the LR and the right image shows the ground truth HR image. The left most image in the lower row shows the result obtained using bicubic interpolation. The middle image in the lower row shows results obtained using sparse representation [4]. The right most image in the lower row shows results obtained by our method. The black rectangular box indicates one of the regions of interest for the viewers to focus. It has been zoomed-in and showed adjacent to the top of the right edge of the images.

SVR Model Selection. We use non-linear SVR to learn our regression model. We used LIBSVM for non linear SVR training [14]. The kernel we used was again a Gaussian kernel $k_s(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|^2)$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{18}$. For choosing the optimum set of regularization parameter C , kernel width parameter γ and ϵ parameter for the ϵ -SVR, we performed grid search using 3 validation images. C was varied as $\{2^0, 2^2, 2^4, \dots, 2^{16}\}$ while γ and ϵ parameter were varied as $\{2^{-6}, 2^{-5}, 2^{-4}, \dots, 2^3\}$ and $\{0.1, 0.2, 0.4, 0.8\}$.

Prediction. When a new LR image is presented for prediction, we up-scale (scaling factor 2) it using bi-cubic interpolation and also take the gradient of the up-scaled image. For each location of the image, we extract 3×3 patch around the location, from the up-scaled image and from its gradient image too. These two patches are vectorized and concatenated to obtain an 18 dimensional vector. Kernel-PCA mapping obtained using 1900 examples (please refer Sect. 2.3) is then used to calculate the new 18 dimensional feature vector which was eventually given as input to the SVR model (learnt using 90000 transformed training examples, refer Sect. 2.3) for prediction (please see Fig. 1).

3 Experiments and Results

We tested our method on 13 unseen test images. These images were neither part of the training images nor part of the validation images. We compared our

Fig. 3. Comparison of results obtained for the “Old Man” image. Left image in the upper row shows the LR and the right image shows the ground truth HR image. The left most image in the lower row shows the result obtained using bicubic interpolation. The middle image in the lower row shows results obtained using sparse representation [4]. The right most image in the lower row shows results obtained by our method. The black rectangular box indicates one of the regions of interest for the viewers to focus. It has been zoomed-in and showed adjacent to the top of the right edge of the images.

Fig. 4. Comparison of results obtained for the “Bike” image. Left image in the upper row shows the LR and the right image shows the ground truth HR image. The left most image in the lower row shows the result obtained using bicubic interpolation. The middle image in the lower row shows results obtained using sparse representation [4]. The right most image in the lower row shows results obtained by our method. The black rectangular box indicates one of the regions of interest for the viewers to focus. It has been zoomed-in and showed adjacent to the top of the right edge of the images.

Table 1. Comparison of results achieved by bi-cubic interpolation, sparse representation method [4] and our proposed method. Results are in dB.

Sr. No.	Image name	PSNR- bicubic	PSNR-[4]	PSNR- Proposed method
1	Hat	31.73	34.02	34.86
2	Parthenon	28.11	29.35	30.33
3	Parrot	31.38	33.92	34.92
4	Butterfly	27.46	31.23	32.60
5	Flower	30.45	32.83	33.76
6	Leaves	27.44	31.38	32.70
7	Bike	25.65	28.18	29.43
8	Baby	30.37	32.45	33.40
9	Chip	32.82	36.96	37.36
10	Pimpled Girl	32.96	35.26	35.66
11	Lena	32.79	35.04	35.86
12	Girl	34.74	35.58	36.18
13	Old Man	32.30	33.94	35.11
	Average	30.63	33.09	34.01

method with one of the state-of-the-art methods [4] (sparse representation based method). For fair comparison we used the same 69 training images as used by [4]. The authors have provided their code [5], as well as their pre-trained dictionary (please refer [4, 5] for details) for the scaling factor of 2. The evaluation was based on Peak Signal to Noise Ratio (PSNR) for the Y channel. The PSNR is calculated as follows:

$$PSNR = 10 \log_{10} \left(\frac{255^2}{E_{ms}} \right) \quad (10)$$

where E_{ms} is the mean squared error. On all the 13 images, our method outperformed the sparse representation based methods [4] as well as the bi-cubic interpolation. The average improvement over sparse representation method [4] was found to be 0.92 dB. Moreover the average improvement over bi-cubic interpolation was found to be 3.38 dB. However, when compared to the bi-cubic interpolation the better PSNR was achieved at the cost of higher execution time. The execution time for our method, for a 128×128 LR image, on a system with 3.40 GHz processor and 8 GB RAM was 225 s. Table 1 shows the results achieved by bi-cubic interpolation, sparse representation method [4] and our proposed method. For visualization purpose we present some of our test LR images with their ground truth HR images, results obtained by bicubic interpolation, sparse representation method [4] and our proposed method in Figs. 2, 3 and 4.

4 Conclusion

We presented a novel method for image super resolution problem. For any categorization task to be successful, we need features having high discrimination

power and also a strong regression/classification method having excellent generalization ability. We deployed two machine learning techniques which exploit the power of kernel-trick. We tested our algorithm on a hold-out data-set of 13 images. Our method outperformed a state-of-the-art method [4]. That we achieved higher PSNR than a state-of-the-art method on each of the 13 test images, shows the generalization power of our method.

References

1. Schölkopf, B., Smola, A.J.: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, Cambridge (2002)
2. Freeman, W.T., Pasztor, E.C., Carmichael, O.T.: Learning low-level vision. *Int. J. Comput. Vision* **40**, 25–47 (2000)
3. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, vol. 1, p. 1. IEEE (2004)
4. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. *IEEE Trans. Image Process.* **19**, 2861–2873 (2010)
5. Yang, J., Wright, J., Huang, T.S., Ma, Y.: (Image Super-resolution via Patch-wise Sparse Recovery <http://www.ifp.illinois.edu/jyang29/ScSR.htm>)
6. Ni, K.S., Nguyen, T.Q.: Image superresolution using support vector regression. *IEEE Trans. Image Process.* **16**, 1596–1610 (2007)
7. An, L., Bhanu, B.: Improved image super-resolution by support vector regression. In: *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pp. 696–700. IEEE (2011)
8. Yuan, T., Yang, W., Zhou, F., Liao, Q.: Single image super-resolution via sparse kpca and regression. In: *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2130–2134. IEEE (2014)
9. Schölkopf, B., Smola, A., Müller, K.R.: Kernel principal component analysis. In: *Artificial Neural Networks ICANN 1997*, pp. 583–588. Springer (1997)
10. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer Science and Business Media, New York (2000)
11. Cortes, C., Vapnik, V.: Support-Vector Networks. *Mach. Learn.* **20**, 273–297 (1995)
12. Weinberger, K.Q., Sha, F., Saul, L.K.: Learning a kernel matrix for nonlinear dimensionality reduction. In: *Proceedings of the Twenty-first International Conference on Machine Learning*, p. 106. ACM (2004)
13. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. *Stat. Comput.* **14**, 199–222 (2004)
14. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. *ACM Trans. Intell. Syst. Technol. (TIST)* **2**, 27 (2011)