Optimized Regressor Forest for Image Super-Resolution

Chia-Yang Chang
cychang@media.ee.ntu.edu.tw
Wei-Chih Tu
wctu@media.ee.ntu.edu.tw
Shao-Yi Chien
sychien@ntu.edu.tw

Media IC and System Lab
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei, Taiwan

Abstract

The goal of image super-resolution is to recover missing high frequency details of an image given single or multiple low-resolution images. It is a well-known ill-posed problem and requires mature prior knowledges or enough examples to restore high-quality high-resolution images. Recently, many methods formulate image super-resolution as a regression problem. Input image patches are classified into pre-trained clusters, and cluster-dependent mapping functions are employed to super-resolve input patches. In this paper, for further improving the reconstructed image quality, an optimized regressor forest framework is proposed, which leverages the discriminative power of random forest. There are three major contributions of the proposed framework. (i) The proposed scheme overturns existing approaches by training the regressors first and learning the way to find the best regressor to avoid quality degradation introduced from the classification outliers. (ii) We propose to employ EM-algorithm to optimize regressors by jointly optimizing the clustering results as well as the regression functions. (iii) In order to find the most appropriate regressor for an input patch at the testing stage, random forest is adopted to accurately classify patches into their best clusters (regressors). The experimental results demonstrate that the proposed method generates high-quality high-resolution images and yields state-of-the-art results.

1 Introduction

Single-image super-resolution (SR) aims at recovering a high resolution (HR) image from a single low resolution (LR) input image. It is a well-known ill-posed problem that has been studied over decades and is still an open problem. Recent mainstream approaches [17, 19] of single-image super-resolution formulate the problem as a data representation learning and regression problem. These methods decompose the input image into patches. These patches are then classified into pre-learned clusters. For each cluster, a linear regressor is pre-trained to transform the input LR patches of this cluster into HR patches. Though it usually takes long time to learn good clustering and regression functions from a large-scale image set at the training stage, at testing stage, the search space (*i.e.* the number of clusters) is orders of magnitude smaller compared to early neighbor embedding [1, 3] or internal example-based

DOI: https://dx.doi.org/10.5244/C.30.85

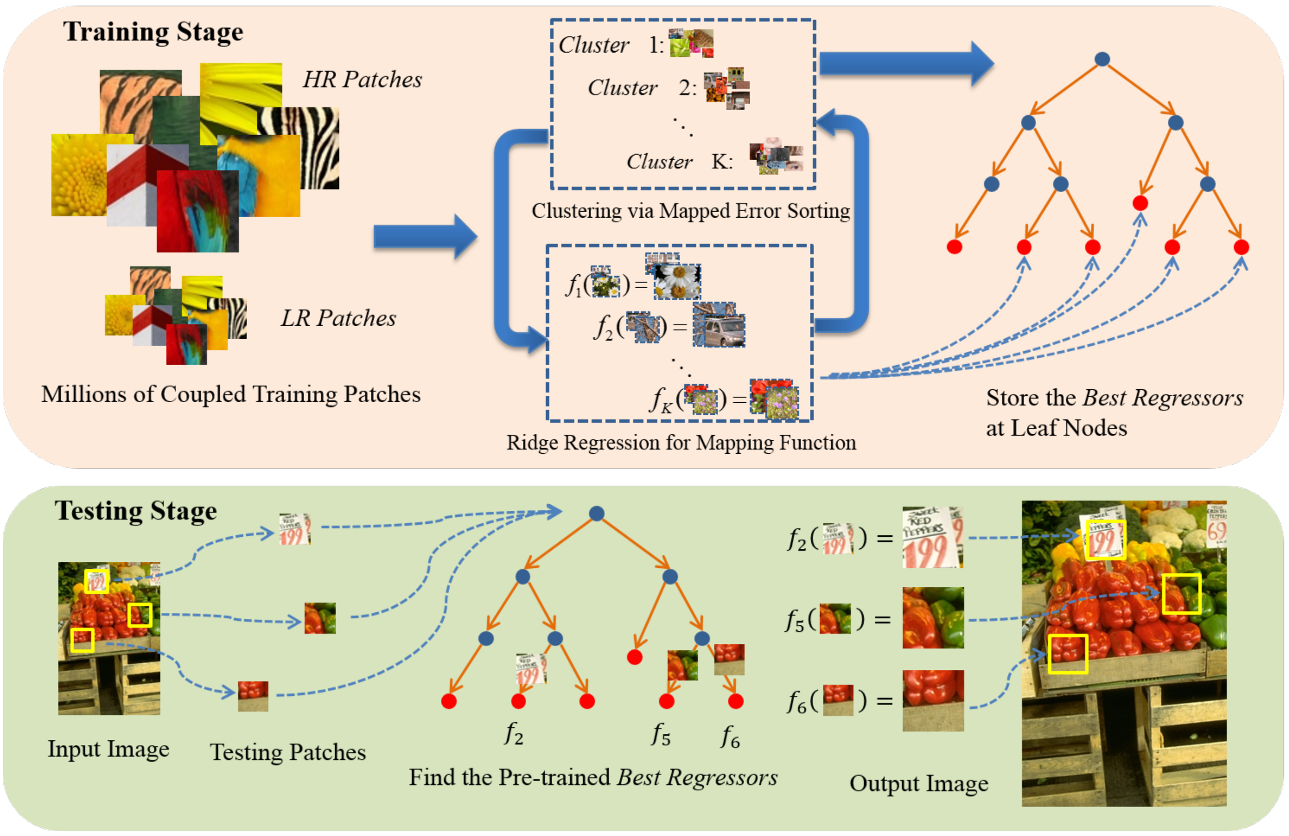

Training Stage
HR Patches
LR Patches
Millions of Coupled Training Patches
Cluster 1:
Cluster 2:
Cluster K:
Clustering via Mapped Error Sorting
Ridge Regression for Mapping Function
Store the Best Regressors at Leaf Nodes
Testing Stage
Input Image
Testing Patches
Find the Pre-trained Best Regressors
Output Image

Figure 1: An overview of the proposed *Optimized Regressor Forest* framework for super-resolution.

SR methods [7, 8, 15]. As a result, the computation of the classification-regression scheme at the testing stage is merely to classify the input patches and to compute the cluster-dependent regression. This kind of classification-regression algorithms achieves current state-of-the-art in single image super-resolution in terms of accuracy and efficiency.

The key towards high quality and efficient image super-resolution turns to answering the following questions:
(i) *How to classify image patches such that patches in the same cluster can be accurately super-resolved using the same regressor?*
(ii) *How to learn the best regression function such that the super-resolved HR patches have highest numerical accuracy?*

Many classification approaches [9, 13, 14, 16, 17, 18, 19, 24] have been explored to tackle the first item and the regression function is learned from the classified patches in the same cluster. The classification following the regression training scheme has a potential problem that the regression step minimizes the error within a cluster. While there might be outliers in the classification process, the learned regression function may not be the optimal for all patches.

In this paper, we tackle the problem in a reverse manner. We put the regressor at the first place. We propose to learn a set of regression functions from training samples using the EM-algorithm. These regressors are learned to minimize overall reconstruction distortion (*e.g.* peak signal to noise ratio(PSNR) or structure similarity(SSIM)) for all training data. After that, we will obtain a set of patch-regressor pairs. We then train a random forest from these patch-regressor pairs to predict the best regressors for input patches. We call it the *Optimized Regressor Forest* for super-resolution. An overview of the proposed framework is illustrated in Fig. 1. Our experimental results show that the proposed method is able to achieve comparable or better results in numerical evaluation or visual comparison.

The rest of this paper is organized as follows. Section 2 reviews related works close to the prosed method. We introduce the proposed framework in Section 3. We show our results and compare with the state-of-the-art approaches in Section 4. Finally, Section 5 concludes this paper.

2 Related Works

The well known bicubic interpolation [10] can be regarded as a special case of the classification-regression scheme where all patches belong to the same cluster and the corresponding regression function is the bicubic kernel. Typically, it has acceptable results in smooth regions while it usually fails to handle edges and textures. The general image-level mapping from a LR image to a HR image is highly complicated such that using only the bicubic kernel can not fully handle all kinds of patches.

The neighbor embedding (NE) approaches [1, 3] are also the milestone of the advances in super-resolution. The input LR patches are approximated by a linear combination of their nearest neighbors in the external dataset. The same combination coefficients are used to fuse the corresponding HR patches in the dataset to predict the HR output patches. Typically, it requires a huge dataset (millions of patches) to achieve decent HR prediction. This also dramatically increases the processing time. Instead of using the patches directly extracted from natural images, Yang *et al.* [22] employed sparse coding to represent the huge image dataset efficiently. Specifically, they jointly learned the compact LR and HR dictionaries with a sparsity constraint using the following sparse representation:

$$\hat{D_h}, \hat{D_l} = \underset{D_h, D_l}{\operatorname{argmin}} \|X_h - D_h\alpha\|_2^2 + \|X_l - D_l\alpha\|_2^2 + \lambda \|\alpha\|_0, \tag{1}$$

where LR patches are denoted by X_l and their corresponding HR patches are denoted by X_h. D_h and D_l are the HR and LR dictionary respectively. α represents the sparse coefficient for both LR and HR patches. At testing stage, the algorithm searches for a sparse representation of each input patch (y_l) as a combination of dictionary atoms:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \|y_l - D_l\alpha\|_2^2 + \lambda \|\alpha\|_0. \tag{2}$$

Then the output HR patch is obtained from $D_h\alpha$, which is a linear combination of the HR dictionary using the same coefficient. The ideal regularization term of sparse constraint is l_0−norm, which leads to an NP-hard problem. Alternatively, Yang *et al.* relaxed it to l_1−norm for feasibility. Zeyde *et al.* [23], build on this work and reach significant improvements both in speed and output quality by using K-SVD [12] to approximate the l_0−norm.

The neighbor embedding approaches may be still computationally expensive due to the optimization of Eq. 2 at testing time. Timofte *et al.* proposed the anchored neighborhood regression (ANR) [16] that relaxes the sparse decomposition optimization of [22, 23] to a ridge regression which can be solved offline and stored at each dictionary atom (anchor). The offline learning scheme enables large speed-up at testing time. Timofte *et al.* further extended the ANR approach to the A+ approach [17, 18]. They learned the regressors from all the training patches in the local neighborhood of the anchoring atoms instead of only from the anchoring atoms as ANR does. Formally, the K-nearest neighbors of each LR dictionary atom (d_i) in LR-HR training patch pairs (N_l, N_h) are collected, then the following ridge regression is computed for each atom.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \|d_i - N_l\beta\|_2^2 + \lambda \|\beta\|_2. \tag{3}$$

This ridge regression has a close-form solution: $\hat{\beta} = (N_l^T N_l + \lambda I)^{-1} N_l^T$. The projection matrix can be precomputed as a mapping function for an input patch y_l:

$$\tilde{y_h} = f_j(y_l) = P_j y_l = N_h (N_l^T N_l + \lambda I)^{-1} N_l^T y_l. \tag{4}$$

The $\tilde{y}_h$ is the super-resolved result, and f_j is the stored mapping function of the atom having the highest correlation to the input patch y_l. The testing time operations of the A+ approach then becomes the nearest-neighbor search followed by a matrix multiplication for each input patch. The ANR and A+ approaches can be understood by the aforementioned classification-regression scheme where they classify patches by finding the nearest neighbors in a set of atoms (anchors) and use the stored regressors to super-resolve input patches.

Some recent SR approaches also explore the same classification-regression scheme with different classification methods. To name a few, Yang *et al*. [19] used K-means clustering to harvest a set of representative patches from natural images and simple affine transform is used as the cluster-dependent regressors for each cluster. Schulter *et al*. [14] adopted the random forest as the classifier and the regressors are learned from the patches in the same leaf node. With the same number of regressors, these methods can perform on par with the A+ method in accuracy.

Note that all above classification-regression methods suffer from a potential problem that the regressors are trained to minimize the prediction error within a cluster. The regression error of each cluster is minimized but the overall regression error for all training data is not guaranteed since the regression might be affected by classification outliers. To address this problem, Dai *et al*. [5] proposed to jointly learn a collection of regressors which yield the smallest super-resolving error for all training data. After learning the *best regressors*, a collection of patch-regressor pairs serves as the transform recipes in testing stage. During testing, the algorithm finds the nearest neighbor for an input patch in the collection of training patches and uses the matched patch's *best regressor* for testing. Due to large size of the search space, Dai *et al*. proposed to use KD-tree to approximate the nearest-neighbor search, which also degrades the accuracy of their method. As reported by [14], the performance of [5] is slightly worse than the A+ method.

We also tackle the problem by first learning the optimal regressors for all training patches such that these regressors yield the minimal error in LR to HR mapping. Then we leverage the discriminative power of random forest to learn the prediction of *best regressors* from input patches. Different from [14], we use the random forest to predict the pre-learned regressors while [14] employs random forest to classify patches and then learns the regressors with the classified patches.The experimental results show that the proposed framework is more effective to predict the *best regressors* that yields smaller super-resolving error compared to the A+ method.

3 Proposed Algorithm

In this section, we present the proposed *Optimized Regressor Forest (ORF)* for super-resolution in detail, including (i) EM-algorithm for best regressor learning and (ii) random forest prediction. An overview of our algorithm is shown in Fig. 1.

3.1 EM-algorithm for best regressor learning

Given the number of regressors, K, we start from the regressors obtained by the A+ algorithm [17]. We aim at learning the best regressors for all training patches. Specifically, an objective function associating with the overall super-resolving error for all training patches is introduced:

$$\hat{S}, \hat{f}_1 \ldots \hat{f}_K = \underset{S, f_1 \cdots f_K}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{K} s_{i,j} \left\| e(x_i^h, f_j(x_i^l)) \right\|_2^2, s.t. \forall i : \sum_{j=1}^{K} s_{i,j} = 1, s_{i,j} \in \{0,1\} \tag{5}$$

Here, N is the number of training patches, x_i^l and x_i^h represent the i-th LR and HR patches in the training set X_l and X_h. f_j denotes the j-th regressor. $e(p,q)$ is a metric measuring the appearance similarity of two patches p and q (*e.g.* MSE or variance). $S \in \mathbb{R}^{N \times K}$ is an indicator matrix. Its element $s_{ij} \in \{0,1\}$ indicates whether the i-th training data has minimal error with the j-th regressor. There is only one s_{ij} being assigned 1 for each i (the one indicating the *best regressor*).

Note that minimizing Eq. 5 not only pairs the optimal regressors to all training data but also adjusts the regressors to reduce the overall error. This minimization is a chicken-egg problem. We tackle it by an EM-algorithm like strategy.

s sub-problem:
Firstly, we consider the problem to find the best regressors given a fixed set of regressors $f_1 \ldots f_K$. This problem can be solved by trial-and-error through all K regressors for each training patch. The one with minimal super-resolving error is assigned to the input patch. This step is equivalent to the M-step of the EM algorithm, which maximizes the posterior probability.

f sub-problem:
After obtaining the indicator matrix S, we consider another problem that S is fixed and the regressors are updated to minimize Eq. 5. Let C_j^l and C_j^h be the gathered training patches which assign the j-th regressor as their best regressor by S. Then the following ridge regression is used to update the K regressors.

$$P_j = C_j^h ({C_j^l}^T C_j^l + \lambda I)^{-1} {C_j^l}^T. \tag{6}$$

This is analogous to the E-step of EM algorithm, that expects the maximum of the posterior probability.

To minimize Eq. 5, we iteratively solve the above two sub-problems. The learning algorithm is detailed in Algorithm 1. We found that 5 iterations are sufficient to achieve decent accuracy.

3.2 Random Forest Reformulation

After minimizing Eq. 5, we have obtained the LR training samples and the updated optimal regressors. Each training sample has been associated with a label indicating its *best regressor*, the one yielding the smallest super-resolving error. We aim to pick the most appropriate regressor for an input patch at the testing stage. Based on the assumption that similar patches can be super-resolved by the same regressor, we formulate the super-resolution as a multi-lable classification problem and train a random forest [2, 4] to predict the best regressors for the input LR patches.

A random forest is an ensemble of decision trees. Each tree consists of split nodes and terminal nodes (leaf nodes). At testing stage, a testing data y (in our case, a LR patch) is passed into the root node. At each split node m, a split function $Split(y, \theta_m)$ is evaluated.

Algorithm 1 EM-algorithm for best regression function training

```
Input: $\{x_i^l, x_i^h\}_{i=1}^N$: training LR-HR patch pairs, $T$: number of iterations
Input: $K$: number of regressors, $e()$: similarity metric
Output: $S = \{s_{ij}\} \in \mathbb{R}^{N \times K}$: indicators of best regressors
Output: $f_1 \dots f_K$: K best regressors
 1: Initialize $f_1 \dots f_K$ by the A+ algorithm [17]
 2: error matrix $O = (o_{ij}) \in \mathbb{R}^{N \times K}$
 3: for iteration $t = 1 : T$ do
 4:     $o_{ij} = e(x_i^h, f_j(x_i^l))$                                ▷ Solving the s sub-problem
 5:     $s_{ij} = \begin{cases} 1, & o_{ij} < o_{ip}, \quad \forall p \neq j \\ 0, & \text{others} \end{cases};$
 6:     for $j = 1 : K$ do                                           ▷ Solving the f sub-problem
 7:         Gather patch pairs $C_j^l$, $C_j^h$ from $X_l, X_h$ using $S$
 8:         $f_j = C_j^h ({C_j^l}^T C_j^l + \lambda I)^{-1} {C_j^l}^T$                 ▷ Ridge regression in Eq. 6
 9:     end for
10: end for
```

This is a binary decision based on some pre-defined features of the input data by thresholding with a learned parameter θ_m. Depending on the decision, the data is passed to the left or right child until a leaf node is reached. At a leaf node, the output is a stored histogram over class labels (in our case, the probability distribution for all regressors).

Each tree in the forest is trained independently. A set of sample LR patches and their corresponding HR patches are provided. We adopted bootstrap aggregation (Bagging) strategy to randomize the training data for each tree. In other words, the training data of each tree is a subset of all training samples, and the probability distribution of all labels (regressors) predicted by each tree is aggregated. The one with the highest aggregated probability will be used as the final regressor. Typically, the more decision trees in the random forest, the overall performance is more stable and accurate.

In our case, the split function is based on thresholding the patch features. We adopt the same patch features as [16, 17, 23]. The features are computed from the first and second order gradients and PCA dimensionality reduction. To learn the parameter θ_m for each split node, we start from the root node, a set of candidate parameters θ are proposed at random. For each candidate, the training set is partitioned into left and right child sets and the following objective function (the typical information gain for classification problem) is evaluated:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} N_L \cdot \sigma(D_L) + N_R \cdot \sigma(D_R), \tag{7}$$

where $\sigma(.)$ denotes the Gini impurity for typical classification framework. D_L and D_R represent the labels of data in L and R respectively. N_L and N_R are the number of samples in the left and right child sets. The typical impurity function $\sigma(.)$ can be expressed as:

$$\sigma(f) = \sum_{j=1}^{K} h_j(1 - h_j), \tag{8}$$

where K is the number of labels, and h_j is the percentage of the data with jth label inn the same child set. After evaluating all candidates, the best parameter respect to Eq. 7 is chosen as the threshold in the split node. As $\sigma(.)$ of a child node is smaller than a user-specified threshold ε or the maximum tree depth is reached, a leaf node is created and the label probability distribution is stored in the leaf node.

In order to reduce the maximum tree depth and maintain the same quality, we reformulate the impurity function $\sigma(.)$ for finding the best threshold. For our regression framework, sometimes the second or third best regressor still provides pleasing result. Instead of impurity, we use the accumulated reconstruction error as follows:

$$\sigma(f) = \sum_{i=1}^{N} \sum_{j=1}^{K} h_j e(x_i^h, f_j(x_i^l)), \tag{9}$$

where $e(.)$ and f_j are with the same definition in Eq. 5, and x_i^l and x_i^h are the samples in child node.

3.3 Testing stage

At the testing stage, the input image is up-scaled by bicubic interpolation first. The bicubic interpolated image is a coarse estimation. We then decompose this image into overlapping patches and compute LR patch features for each patch. As mentioned earlier, we adopt the same LR features as [16, 17, 23]. Next, each LR patch is passed into the random forest. There are two ways to determine the regressor for each input LR patch from the probability distribution of regressors stored at leaf nodes. The first way is to select the best regressor with the highest probability. The other way is to aggregate the HR patches obtained by the regressors at each leaf node. Among these two approaches, the latter gives better quality since it can derive the expected value of the reconstructed HR patch.

4 Experimental Result

In this section, we evaluate the performance of the proposed ORF framework for image super-resolution and give the numerical evaluation of the proposed framework.

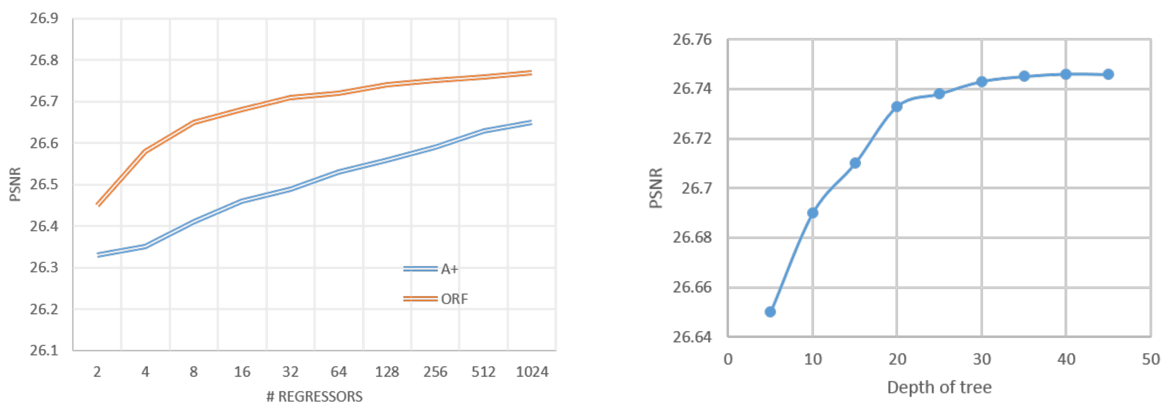
We use the 91 training images as proposed by Yang *et al*. [21]. We work only on the luminance component in YCbCr color space, and the chroma components are bicubically interpolated as previous works do. Gaussian kernel is employed to remove high frequency details from HR images, which are then sub-sampled to obtain LR images. The degradation operators are different from bicubic down-sample, since these operators are close to camera model. The results of compared methods would also be slightly difference from original papers. The testing sets contain the standard super-resolution benchmarks **Set 5**, **Set14** and 100 images from **Berkeley Segmentation Dataset** (BSD) [11]. The testing images are also blurred by Gaussian kernel and sub-sampled. We treat this testing as non-blind super-resolution [20], that is, the Gaussian kernel used for training and testing images are the same.

4.1 Patch features

The type of features used to represent image patches is an important factor that can influence the performance. A popularly used and robust feature is the first and second order derivatives of the patch [3, 23]. We adopt the same feature as those of [16, 17, 23], which start from the first and second order gradients and apply PCA to reduce dimensionality. This patch feature is used in regressor learning as well as the splitting criteria learning in the random forest.

Table 1: Results of the proposed method compared with state-of-the-art works on 3 datasets using three different magnification factors.

		Bicubic	Zeyde	NE+NNLS	NE+LLE
dataset	factor	PSNR/SSIM/Time(s)	PSNR/SSIM/Time(s)	PSNR/SSIM/Time(s)	PSNR/SSIM/Time(s)
Set5	x2	27.74/0.85981/0.0000	30.37/0.91211/0.4552	31.25/0.92625/14.1358	31.47/0.92922/1.7616
	x3	27.22/0.84375/0.0000	29.06/0.88643/0.3258	29.11/0.88879/7.2368	29.55/0.89833/0.9024
	x4	26.44/0.81578/0.0000	27.59/0.84799/0.2506	27.59/0.84943/3.4766	27.83/0.85626/0.6107
Set14	x2	25.43/0.76590/0.0000	27.33/0.83483/0.9740	28.09/0.86155/28.7368	28.26/0.86633/3.7238
	x3	24.99/0.74413/0.0000	26.28/0.79760/0.6049	26.28/0.80140/14.3354	26.66/0.83450/1.7791
	x4	24.34/0.71115/0.0000	25.10/0.74842/0.4822	25.09/0.74999/6.6289	25.25/0.76065/1.1338
B100	x2	25.26/0.73056/0.0000	26.58/0.79912/0.7033	27.23/0.83012/19.3014	27.34/0.83462/2.4847
	x3	24.88/0.70749/0.0000	25.74/0.75960/0.4179	26.00/0.77730/9.4292	26.08/0.78310/1.2700
	x4	24.33/0.67400/0.0000	24.82/0.71083/0.3852	24.82/0.71281/4.9159	24.95/0.72493/0.8543
		ANR	A+	ORF-PSNR	ORF-SSIM
dataset	factor	PSNR /SSIM /Time(s)	PSNR/SSIM/Time(s)	PSNR / SSIM / Time(s)	PSNR/SSIM/Time(s)
Set5	x2	31.40/0.92814/0.3839	32.50/0.94068/0.3839	**32.95**/0.94347/0.6556	32.88/**0.94416**/0.6432
	x3	29.60/0.89878/0.2649	30.54/0.91561/0.2649	**30.83**/**0.92034**/0.4079	30.81/0.92033/0.4125
	x4	27.87/0.85727/0.2294	28.51/0.87519/0.2294	**28.64**/0.87840/0.3633	28.62/**0.87846**/0.3585
Set14	x2	28.20/0.86424/0.8479	28.83/0.86984/0.8479	**29.28**/0.87895/1.3888	29.22/**0.87965**/1.4012
	x3	26.70/0.81760/0.5153	27.38/0.83751/0.5153	**27.63**/0.83162/0.7898	27.62/**0.83783**/0.7632
	x4	25.28/0.76162/0.4370	25.72/0.75975/0.4370	**25.88**/0.77700/0.6856	**25.88**/**0.77709**/0.6901
B100	x2	27.29/0.83240/0.5937	27.83/0.84123/0.5937	28.18/0.85161/0.9627	**28.19**/**0.85225**/0.9523
	x3	26.09/0.78310/0.3597	26.56/0.79799/0.3597	**26.74**/0.80283/0.5550	**26.74**/**0.80300**/0.5732
	x4	24.96/0.72587/0.3517	25.22/0.72005/0.3517	**25.34**/0.73653/0.5357	**25.34**/**0.73671**/0.5052



(a) PSNR to number of regressors (b) PSNR to depth of 15 trees

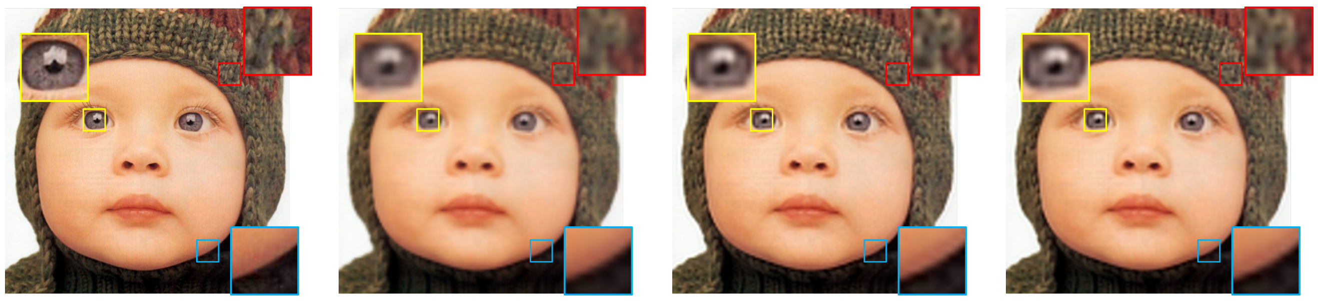
Figure 2: PSNR analysis on B100 with factor x3.

4.2 Compared methods

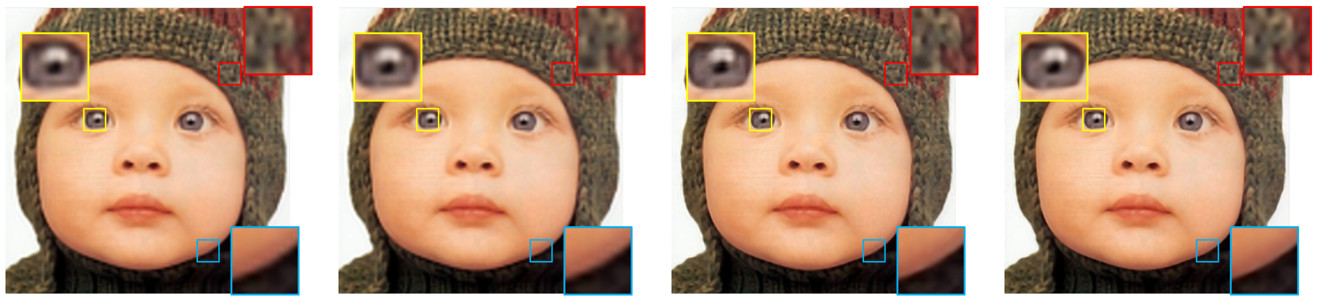
We compare the proposed ORF framework with six classical or state-of-the-art methods[1]. They are bicubic interpolation [10], neighbor embedding with locally linear embedding (NE+LLE) [3], neighbor embedding with non-negative least squares (NE+NNLS) [1], sparse representation SR (Zeyde) [23], anchored neighborhood regression (ANR) [16] and A+ [17]. To fairly compare these methods, all of them including ours use the same patch features. For learning based methods, the training set is the same as [21] and the number of regressors is fixed to 128 in our experiments. The furthermore analysis of the number of regressors could be found in Fig. 2(a)

We set the number of trees in the random forest as 15, and the maximum depth of each tree is 30. Although using more trees and allowing deeper tree depths gives more accurate results, the parameters are set to prevent over-fitting. See the detail analysis in Fig. 2(b). We

[1]We use the MATLAB code provided by Timofte *et al*. [17]. All the compared methods are included in `http://www.vision.ee.ethz.ch/~timofter/ACCV2014_ID820_SUPPLEMENTARY/index.html`



(a) Ground Truth (b) Bicubic (c) NE+LLE (d) NE+NNLS



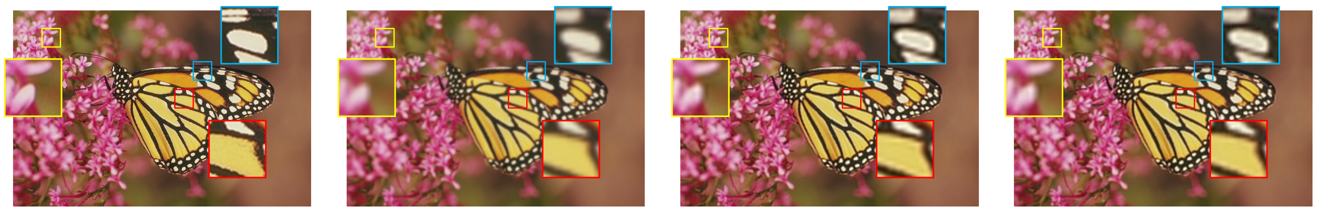
(e) Zeyde (f) ANR (g) A+ (h) Our ORF-PSNR

Figure 3: Baby image from Set5 dataset with upscaling 4x. The results show ours and A+ have more details, but A+ has slightly ringing effects.

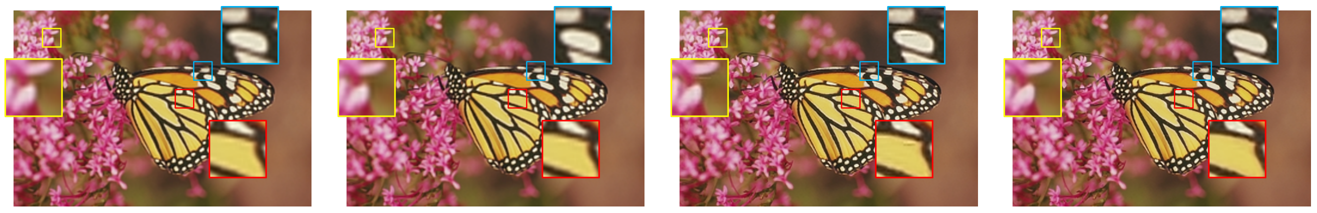
have tried two versions of $e(.)$ in Eq. 5. One is the MSE metric, which is called as *ORF-PSNR* since it minimizes Eq. 5 to maximize PSNR. Another version of $e(.)$ uses the structure similarity (SSIM), and we call it *ORF-SSIM* since it maximizes SSIM.

Table 1 summarizes the numerical evaluation results, where the average PSNR and SSIM scores for different datasets and magnification factors are shown. Our ORF framework achieves comparable or better accuracy in both PSNR and SSIM evaluation. Figs. 3, 4, and 5 show some super-resolved results for visual comparison. Compared to other methods, the proposed ORF framework has better capability to recover missing details.

Comparing our ORF to other methods, the improved visual quality of our result is obvious. Moreover, the objective quality metrics PSNR and SSIM support this result. We attribute the better performance to the EM-algorithm and accurate classifier.



(a) Ground Truth (b) Bicubic (c) NE+LLE (d) NE+NNLS



(e) Zeyde (f) ANR (g) A+ (h) Our ORF-PSNR

Figure 4: Butterfly image from Set14 dataset with upscaling 3x. The ringing effects aroud egde still happen in A+. We assume that is caused by non-optimal regressors.

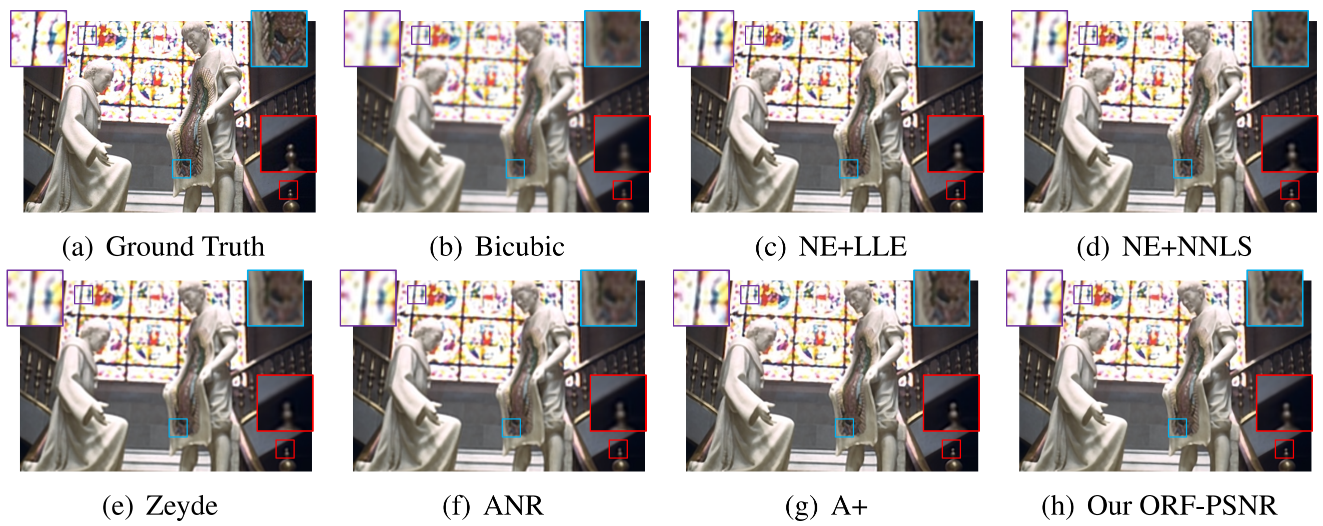


Figure 5: Statue image from BSD dataset with upscaling 2x. 2x is the simple case for all the methods, but our improvements are still easily noticeable

5 Conclusion

In this paper we propose a super resolution framework based on the optimized regressors and random forest. The optimized regressors could preserve the custom-define feature, and we have shown that in the cases of mean square error and structure similarity. With the power of random forest, the best regressor could be found no matter what features the regressors are optimized for. Experimental results show that the proposed algorithm outperforms the other methods both in PSNR and SSIM. In future work, we will try to extend our method to improve the other computer vision works [6] to be regardless of input resolution.

Acknowledgement

This work was supported by Himax Technologies, Inc.

References

[1] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *British Machine Vision Conference*, 2012.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[4] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

[5] Dengxin Dai, Radu Timofte, and L Van Gool. Jointly optimized regressors for image super-resolution. *Computer Graphics Forum*, 34(2):95–104, 2015.

[6] Dengxin Dai, Yujian Wang, Yuhua Chen, and Luc Van Gool. Is image super-resolution helpful for other vision tasks? In *IEEE Winter Conference on Applications of Computer Vision*, volume 6, 2016.

[7] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *IEEE International Conference on Computer Vision*, pages 349–356, 2009.

[8] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2015.

[9] Xinwei Jiang, Jie Yang, Lei Ma, and Yiping Yang. Multi-task Gaussian process regression-based image super resolution. In *British Machine Vision Conference*, 2015.

[10] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981.

[11] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, volume 2, pages 416–423, 2001.

[12] Ron Rubinstein, Tomer Peleg, and Michael Elad. Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model. *IEEE Transactions on Image Processing*, 61(3):661–677, 2013.

[13] J. Salvador and E. Pérez-Pellitero. Naive Bayes Super-Resolution Forest. In *IEEE International Conference on Computer Vision*, pages 325–333, 2015.

[14] Samuel Schulter, Christian Leistner, and Horst Bischof. Fast and accurate image upscaling with super-resolution forests. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3799, 2015.

[15] Abhishek Singh and Narendra Ahuja. Super-resolution using sub-band self-similarity. In *Asian Conference on Computer Vision*, pages 552–568. 2014.

[16] Radu Timofte, Vincent De Smet, and Luc Gool. Anchored neighborhood regression for fast example-based super-resolution. In *IEEE International Conference on Computer Vision*, pages 1920–1927, 2013.

[17] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian Conference on Computer Vision*, pages 111–126. 2014.

[18] Radu Timofte, Rasmus Rothe, and Luc Van Gool. Seven ways to improve example-based single image super resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 111–126. 2016.

[19] Chih-Yuan Yang and Ming-Hsuan Yang. Fast direct super-resolution by simple functions. In *IEEE International Conference on Computer Vision*, pages 561–568, 2013.

[20] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-image super-resolution: a benchmark. In *European Conference on Computer Vision*, pages 372–386. 2014.

[21] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[22] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[23] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. 2010.

[24] Kaibing Zhang, Dacheng Tao, Xinbo Gao, Xuelong Li, and Zenggang Xiong. Learning multiple linear mappings for efficient single image super-resolution. *IEEE Transactions on Image Processing*, 24(3):846–861, 2015.