Single Image Super-Resolution Using Multiple Extreme Learning Machine Regressors

Daniel Luis Cosmo
Laboratório de Computação
e Sistemas Neurais,
Universidade Federal do Espírito
Santo, Vitória, Espírito Santo.
Email: daniel.cosmo@aluno.ufes.br

Fernando Kentaro Inaba
Laboratório de Computação
e Sistemas Neurais,
Universidade Federal do Espírito
Santo, Vitória, Espírito Santo.
Email: kentaro@ele.ufes.br

Evandro Ottoni Teatini Salles
Laboratório de Computação
e Sistemas Neurais,
Universidade Federal do Espírito
Santo, Vitória, Espírito Santo.
Email: evandro@ele.ufes.br

***Abstract*—This paper presents a new technique to solve the single image super resolution reconstruction problem based on multiple extreme learning machine regressors, called here MELM. The MELM employs a feature space of low resolution images, divided in subspaces, and one regressor is trained for each one. In the training task, we employ a color dataset containing 91 images, with approximately 5.3 million pixels, and PSNR and SSIM as metric evaluation. For the experiments we use two datasets, Set 5 and Set 14, to evaluate the results. We observe MELM improves reconstruction quality in about 0.44 dB PSNR in average for Set 5, when compared with a global ELM regressor (GELM), trained for the entire feature space. The proposed method almost reaches deep learning reconstruction quality, without depending on large datasets and long training times, giving a competitive trade off between performance and computational costs.**

I. Introduction

The purpose of Super-Resolution (SR) algorithms is to estimate a high resolution (HR) image corresponding to a given low resolution (LR) input image. We can estimate a HR image from a set of LR images with sub-pixels shifts or from one unique LR image. The former is known as Multi Frame Super-Resolution and the latter as Single Image Super-Resolution.

The LR image Y is generated from the HR image X using the model

$$Y = (B * X) \downarrow s, \tag{1}$$

where B denotes a blurring filter to prevent aliasing, $\downarrow$ is a downsampling operator with downsampling factor s, and the operator $*$ represents convolution. Recovering X from Y is a ill posed inverse problem, and some sort of regularization is commonly used to solve equation (1).

SR reconstruction algorithms are needed due to hardware limitations on imaging acquisition devices, like the presence of optical distortion, lens and motion blur, and insufficient sensor sampling and aliasing. Image quality on the output of those devices is usually bellow desired, and hardware solutions are impractical. To circumvent this problem, SR reconstruction algorithms are used as software solutions implemented on the imaging systems.

SR reconstruction techniques are present in several fields: medical imaging applications, where results of low density radiation scans are reconstructed using SR techniques to improve image quality without raising the exposure of patients to radiation [1]; Surveillance applications, when applying SR reconstruction in photos or videos, improving image quality in areas of interest, like faces and car plates [2]; Remote Sensing, where one or multiple satellite images with low quality are used to estimate a HR image [3].

The simplest techniques available to provide super-resolution are based on interpolation, which relies on filtering operations between the image and a linear kernel. These approaches are simple and computationally efficient, but the resized output image lacks high frequency information due to low pass behavior of interpolation filters.

Multi frame techniques uses information contained in multiple LR images of the same scene to estimate a HR image. The LR images must have subpixel shifts between each other to offer extra information of the scene, and a registration method is applied to estimate these displacements. Multi frame SR can be broadly categorized in frequency domain [4], non uniform interpolation [5] and Bayesian inference [6], [7] approaches.

Another group of SR techniques are example-based approaches. These techniques use a training set consisting of LR and HR image pairs to learn a model or mapping between them. This model or mapping is used on a desired LR image to estimate the HR image or hallucinate their missing high frequency components. The interest of example based approaches roses with the work [8], where they conducted a per-patch nearest neighbor search in the dataset to find the lost high frequency components of the interpolated LR input image.

Others example-based approaches are: Neighbor Embedding [9]–[12], where reconstruction weights are calculated, based on training samples, for the LR input patch and used to estimate the corresponding HR output patch; Sparse Coding [13]–[16], where an LR and HR dictionaries are jointly trained based on the training samples, and used to estimate the HR patch based on sparse coefficients calculated from the LR patch; Deep Learning [17]–[20], where convolutional neural networks are used to model the mapping between LR an HR patches.

In this paper, we use the example-based approach, proposing

a SR reconstruction algorithm using several single-hidden-layer feed-forward networks (SLFNs) trained with Extreme Learning Machine (ELM) method. Each SLFN is trained using a set of LR and HR example pairs from a dataset, grouped based on the similarity between their feature vectors. We use k-means to create clusters in the feature space, and a single SLFN is trained for each cluster. Recent research [21]–[23] show that training local regressors instead of one global regressor for the entire feature space gives better SR reconstruction. We believe that training a regressor for each cluster makes it more specific for that region of the feature space, estimating a better mapping of the LR e HR patches in that region.

We started this work with the initial idea of a ELM global regressor of [24]. The main contributions of this paper are:

- Adapting the global regressor of [24] to a more refined state, consistent with state of the art SR reconstruction algorithms;
- Training multiple ELM regressors, considering that ELM training is much faster than back-propagation and gives good generalization when used with a regularization term.

The remainder of the paper is organized as follows: in section II we take a look on how ELM works; in section III we explain the global ELM algorithm, in section IV we present our proposed algorithm, in section V we give details of the experiment, in section VI we compare the results of our algorithm with state of the art SR reconstruction methods and in section VII we draw conclusions about this work.

II. Extreme Learning Machine

Extreme Learning Machine is characterized by being a extremely fast learning algorithm for the single-hidden-layer feed-forward networks [25]–[27]. One of the features of ELM is that the hidden layer need not be tuned, and the calculation of the output layer is a closed form solution.

The output function for one node of ELM, for generalized SLFNs, is

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta. \tag{2}$$

where L is the number of neurons on the hidden layer, $\beta = [\beta_1, \ldots, \beta_L]^T$ is the vector of output weights between the hidden layer and the output node and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_L(\mathbf{x})]$ is the output row vector of the hidden layer with respect to the input $\mathbf{x}$.

The goal of ELM learning is to reach not only the smallest training error but also the smallest norm of output weights. According to [28], as the norm of output weights gets smaller, the generalization of the network gets better. The function ELM minimizes is

$$\arg\min_{\beta} = \|\mathbf{H}\beta - \mathbf{Y}_t\|^2 + C\|\beta\|^2, \tag{3}$$

where $\mathbf{Y}_t$ is the output matrix of target values, C is a regularization parameter and $\mathbf{H}$ is the hidden layer output matrix given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \ldots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \ldots & h_L(\mathbf{x}_N) \end{bmatrix}. \tag{4}$$

The solution of the optimization problem in equation (3), for the case where the number of training samples is bigger than the number of hidden neurons, is

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{Y}_t. \tag{5}$$

After obtaining β in (5), the output function of ELM regressor is

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{Y}_t. \tag{6}$$

To reach this result, first we have to calculate the hidden layer output matrix

$$\mathbf{H}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \ldots, G(\mathbf{a}_L, b_L, \mathbf{x})], \tag{7}$$

where $\mathbf{a}_i$ are the weights between the input layer and the hidden layer, b_i are the biases of the hidden layer and $G(\cdot)$ is the activation function of the hidden neurons. The weights and bias are randomly generated according to any continuous probability distribution and the activation function can be any nonlinear piecewise continuous function satisfying ELM universal approximation capability theorem [29].

III. Global ELM regressor for SR reconstruction

The global ELM regressor [24] uses the same idea used by most exampled-based algorithms, that is, using the training dataset to estimate missing high frequency information from an interpolated LR image that we want to reconstruct. The algorithm can be separated in two steps: training and SR reconstruction.

In the training step, a training dataset consisting of HR images I_{HR} is used. From I_{HR} images, low resolution images I_{LR} are generated by convolving I_{HR} with a blur kernel and downsampling, as in equation (1), creating a training dataset corresponding to pairs of LR and HR images. For each LR image, some basic interpolation method is used, forming a initially upscaled LR image I_0, with the same size as I_{HR}. The high frequency components image I_{HF} can be calculated as

$$I_{HF} = I_{HR} - I_0. \tag{8}$$

The goal of ELM regressor is to learn the mapping between the initially upscaled image I_0 and its missing high frequency components I_{HF}. This mapping is done per patch, instead of per image, because it is easier to map small local regions with similar geometry an texture than a full image.

To create the sample vectors $\mathbf{x}$, the input for the SLFN, the image I_0 is transversed in a raster-scan order, and a local neighborhood ($m \times m$) is selected centered at the pixel

location. Features are extracted from the patches (normally based on the local gradients) and ordered lexicographically, forming the matrix of input vectors $\mathbf{X}$. The target matrix $\mathbf{Y}_t$ is formed by the value of pixels in I_{HF}, so for each input patch there is a output pixel. The matrix $\mathbf{X}$ and $\mathbf{Y}_t$ are used to train the ELM regressor.

The SR reconstruction step begins by using the same initial interpolation in the desired image I_{LR} that we want to reconstruct, forming I_0. After obtaining I_0, the same features are extracted and fed to the trained ELM regressor, in the same raster scan order. The outputs of the trained ELM are the pixels containing high frequency information, forming the image I_{HF} when combined. The high resolution image I_{HR} can be calculated as

$$I_{HR} = I_0 + I_{HF}. \tag{9}$$

More detailed information about values and parameters used in the global ELM regressor can be seen in section V.

IV. Multiple ELM regressors for SR reconstruction

Our proposed SR reconstruction algorithm is based on the idea of multiple linear maps presented in [21], [23]. The idea to use ELM regressors comes from the fact that a SLFN regressor can be more robust than simple linear mappings, like the ones adopted in [21], [23]. Also, training multiple SLFN regressors using the ELM methodology is not computationally demanding, given that the training is not iterative and has a closed form solution.

The core idea of multiple maps is to divide the LR feature space and HR space in groups or subspaces, based on a similarity measure, and use one regressor to learn the map for each of these subspaces. This way, each regressor will be more specialized to map samples from the subspace they were trained, in contrast to a global regressor, who maps the entire feature space.

The proposed algorithm can be divided again in two phases: training and SR reconstruction.

A. Training phase

Suppose we have a dataset of LR-HR patch pairs, created from some HR images in the same way explained in section III and shown in Figure 1. Let $\mathbf{X_s} = \{\mathbf{x}_s^i\}_{i=1}^N$ be the LR training patches and $\mathbf{Y}_s = \{\mathbf{y}_s^i\}_{i=1}^N$ be the HF training patches, where N is the number of samples.

In order to divide this dataset in subsets, we apply k-means clustering in the training samples $\mathbf{X_s}$. This leads to K clusters or subsets of $\mathbf{X_s}$ and K centroids $\{\mathbf{c}_k\}_{k=1}^K$. Let $\mathbf{X}_s^k = \{\mathbf{x}_s^i\}_{i\in\Omega_k}$ be the k subset of $\mathbf{X}_s$, where Ω_k are the indexes of samples belonging to cluster k. We use these same indexes to cluster the HF training samples, creating the subsets $\mathbf{Y}_s^k = \{\mathbf{y}_s^i\}_{i\in\Omega_k}$. At the end, we have K coupled subsets of LR-HF samples, $\{\mathbf{X}_s^k, \mathbf{Y}_s^k\}$. We use each coupled subset to learn K models for ELM regressors. The training process can be seen in Figure 2.

B. SR reconstruction phase

Beginning with a LR image that we wish to reconstruct, a series of operations are made: initial upscaling to make the LR image have the same size as the HR target image, feature extraction to better represent the image and a raster-scan sweep to extract a feature patch centered at each pixel of the image. At the end, we have a set os LR patches $\mathbf{X} = \{\mathbf{x}^i\}_{i=1}^{N_i}$, where N_i is the number of pixels in the initially upscaled image I_0.

After extracting the patches from the LR image, we measure the similarity metric (same used in k-means) between each patch $\mathbf{x}^i$ and the set of centroids $\{\mathbf{c}_k\}$ found in the training phase. Each patch is then associated to the nearest cluster, and the ELM regressor trained for this specific cluster is used to reconstruct the patch, generating a high frequency patch $\mathbf{y}^i$. After reconstruction and rearranging all patches $\{\mathbf{y}^i\}_{i=1}^{N_i}$, we have the HF image I_{HF}, which we sum with the initially upscaled image I_0 to obtain the HR output image I_{HR}. The SR reconstruction process can be seen in Figure 3.

V. Experiment

In this section we detail the settings and parameters used in our Multiple ELM algorithm, and compare some changes we made from the work of L. An and B. Bhanu [24], for the global ELM regressor.

A. Color space

The color space we use in our algorithm is YCbCR, and the SR reconstruction was only applied in the Y (luminance) channel. In the last years, the majority of SR works [11], [12], [14], [17], [23], [24] used this same setup, once the human visual system is much more sensitive to high frequency intensity changes than high frequency color changes. To reconstruct RGB images, first we transform the color space to YCbCr, them we apply the SR reconstruction algorithm on the Y channel, while in the other two channels we apply bicubic

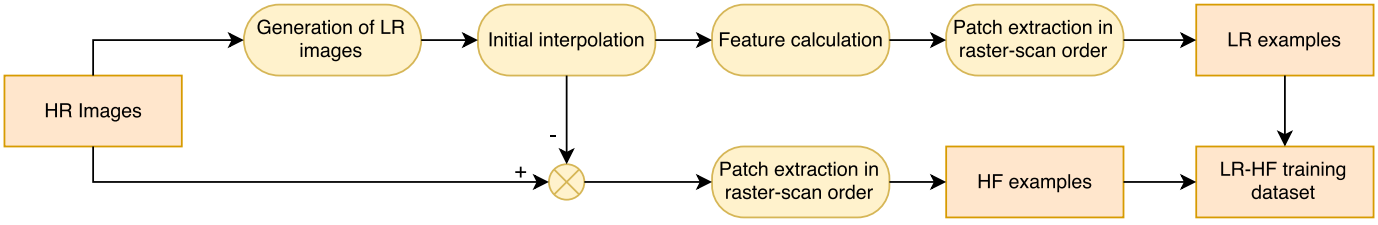


Fig. 1. Flowchart explaining the steps used to construct a dataset of coupled low resolution and high frequency patches.

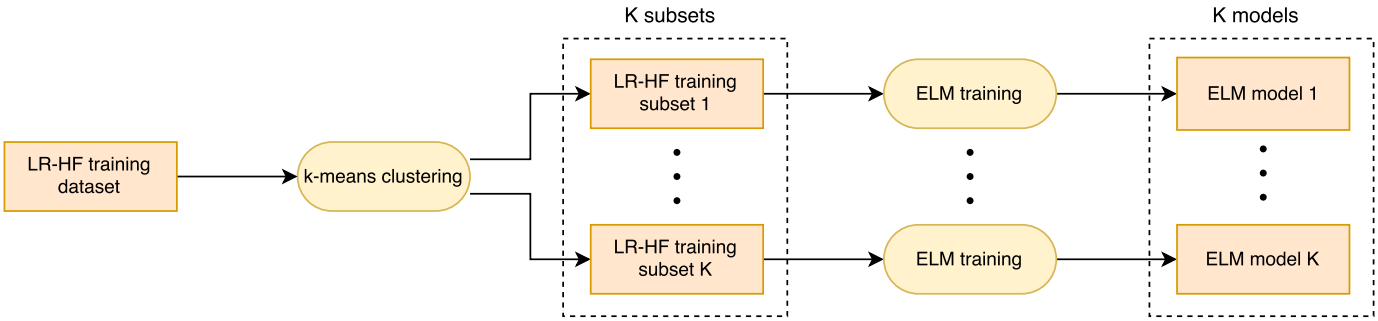


Fig. 2. Flowchart explaining the steps used to learn multiple ELM models form a dataset of coupled LR-HF samples.

interpolation. The final HR image is obtained by transforming back the YCbCr image to RGB space.

B. Features

The features used in [24] are pixel intensity values from local image patches and 1st and 2nd order derivative magnitudes. At each pixel (i, j) of the initial interpolated image I_0, the feature vector is formed by a local patch of pixel intensities centered at pixel (i, j) and five derivative values $\left(\frac{\partial I_0}{\partial \mathbf{x}}, \frac{\partial I_0}{\partial \mathbf{y}}, \frac{\partial I_0}{\partial \mathbf{x^2}}, \frac{\partial I_0}{\partial \mathbf{y^2}}, \frac{\partial I_0}{\partial \mathbf{xy}}\right)$ calculated at position (i, j).

In our work, we use the same features, but instead of using the derivative values calculated only at the central pixel, we use the values calculated in a patch centered at the central pixel. This neighborhood information gives better characterization to the patch, once natural images have similar neighborhood structure. The derivatives are calculated using the filters developed in [30].

We use a patch of size 5×5, extracting 25 values of pixel intensities and 125 derivative values (25 values for each partial derivative). All information extracted from the patch are arranged in a vector of dimension 150. We use PCA to reduce the feature vector to dimension 50, while maintaining 99.9% variance of the features. In our test, this dimensionality reduction don't affect the reconstruction quality. All feature vectors extracted from the training dataset are globally normalized to $[-1, 1]$ before being inputed to the ELM regressor for training.

C. HF output

In [24], the ELM output for an input patch centered at pixel (i, j) is a pixel of the HF image, located at the same (i, j) position. In our work, instead of a output pixel, we have an output patch centered at pixel (i, j), creating overlapping patches in the HR grid. The pixels final values are the average of all contributions given by the patches overlapping that pixel. This way, input pixels located at the neighborhood of position (i, j) contributes to the final value of HF pixel at (i, j). According to empirical results, we chose a 5×5 patch output.

D. Algorithm parameters

Two parameters in the ELM regressor are user specified: number of hidden neurons (L) and regularization weight (C). Performance of ELM is not very sensitive to these parameters, as can be seen in [27], and good results can be achieved as long as L is large enough. In our algorithm, we set $L = 1,000$ and $C = 256$. Sigmoid function is used as activation function of the hidden neurons.

The number of clusters (K) generated by k-means is another specified parameter. When K grows, two situations occur.

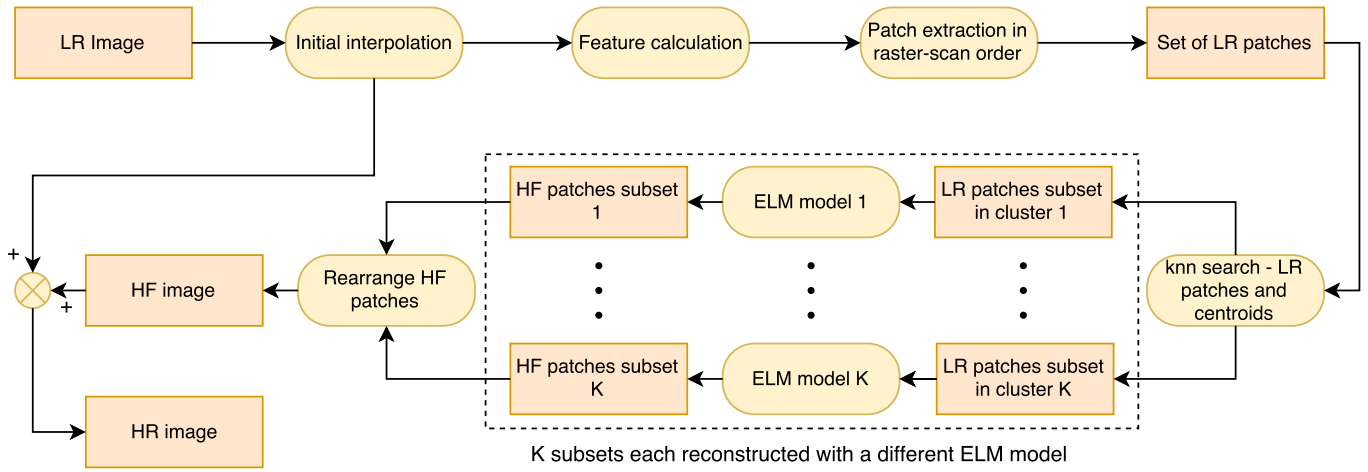


Fig. 3. Flowchart explaining the steps used to SR reconstruct a LR input image.

First, the number of subsets grow, leading to an increase in numbers of more specialized ELM regressors. Second, the number of samples in each subset diminishes, leading to less samples to train each ELM. At the end, K must be chosen in order to maximize the number of cluster without leaving them with a little quantity of samples. In our tests, $K = 50$ gave good results.

E. Training dataset

The dataset we use for training the ELM regressors is the same used in [17], containing 91 images. This dataset contains approximately 5.3 million pixels, leading to the same amount of samples, considering that each pixel leads to a feature vector. Due to memory limitations, we don't use all samples to calculate the clusters with k-means. Instead, 8,000 samples are randomly extracted from each image, after feature calculation, composing a LR-HF dataset with around 728,000 samples (some images are shorter than 8,000 pixels, so the number of samples is a little shorter than 728,000). The choice to use this dataset, instead of a smaller one, as used in [24] (with 5 images), resides in the fact that with more images we can capture samples with more variability, giving better generalization to the trained regressors.

Another memory limitation occurs when we train a ELM regressor for a cluster with too many samples. To alleviate this problem, we set a maximum number of samples per cluster equal to 300,000.

After applying k-means clustering, we associate the remaining samples, that were not used for clustering, to the closest clusters that are not full, using the distance between samples and cluster centroids. After all samples are associated, the regressors are trained.

VI. Results and analysis

In this section, we compare our work with 3 state of the art methods:

- SC - Sparse Coding super resolution method [14],
- ANR - Anchored Neighborhood Regression method [11],
- CNN - Convolutional Neural Network super resolution method [17],

along with bicubic interpolation as a baseline.

The implementations of all methods above are publicly available, and all algorithms runs in the same computer, with an i5 2500 processor and 8 GB of RAM memory. We use the same bicubic kernel to downsample the HR images. For all algorithms, except SC, we use the models trained by the authors. We train a model for SC because the model available was trained using a gaussian kernel to downsample the images. We train this model using the same proposed parameters reported by the authors. We don't compare directly with [24] because we couldn't find any publicly algorithm available from the authors.

We use two datasets to compare all methods:

- Set 5 [10], with 5 images,
- Set 14 [15], with 14 images.

We run all methods in 3 upscales, $\times 2$, $\times 3$ and $\times 4$. Set 5 is used to validate our algorithm parameters with $\times 2$ upscale. The metrics used to compare results are PSNR and SSIM [31]. Those metrics are used in almost all recent super-resolution works, and are calculated using only the luminance channel. Reconstruction time is also reported for each method. We don't cut borders of the reconstructed images before calculating the metrics, thus some results can be slight different from the original papers.

We use two versions of our algorithm. In the first version, denominated GELM (Global ELM), we train only one ELM regressor for the entirety of the feature space. This version is In the second approach, denominated MELM (Multiple ELM), we first apply k-means and them train one ELM regressor for each subspace. A comparison of all the results can be seen in in Tables I and II. All results are average values across the specified dataset. Figures 4 and 5 shows some reconstructed images of set 14 for all methods, except GELM, with $\times 4$ upscale.

Results show that the GELM method is comparable with SC and ANR, but its reconstruction quality is below CNN. MELM methods improve PSNR and SSIM when compared to GELM, almost reaching the results given by CNN method, without increasing reconstruction time. This happens because the same amount of patches needs to be reconstructed in both GELM and MELM, not depending on the quantity of ELM regressors used.

MELM results stays somewhat below deep learning results, but the former don't need huge training datasets and don't need much time to train. As was stated in [17], the CNN model used was trained using a dataset with 395,909 images and took 8×10^8 back-propagations to train (training time was not stated in the paper). Our MELM method uses a dataset with 91 images and take less than 10 minutes to train.

VII. Conclusion

In this paper we proposed a learning based method to single image super resolution. The proposed method uses a efficient and fast training method for single-hidden-layer feed-forward networks called extreme learning machine, to learn the mapping between low resolution and high frequency regions of images. Super-resolution reconstruction quality is further improved by dividing the feature space of the training samples in clusters, with k-means, and training one extreme learning machine regressor for each cluster. Results show that the proposed method almost reaches deep learning reconstruction quality, without the need of large image datasets or long training times.

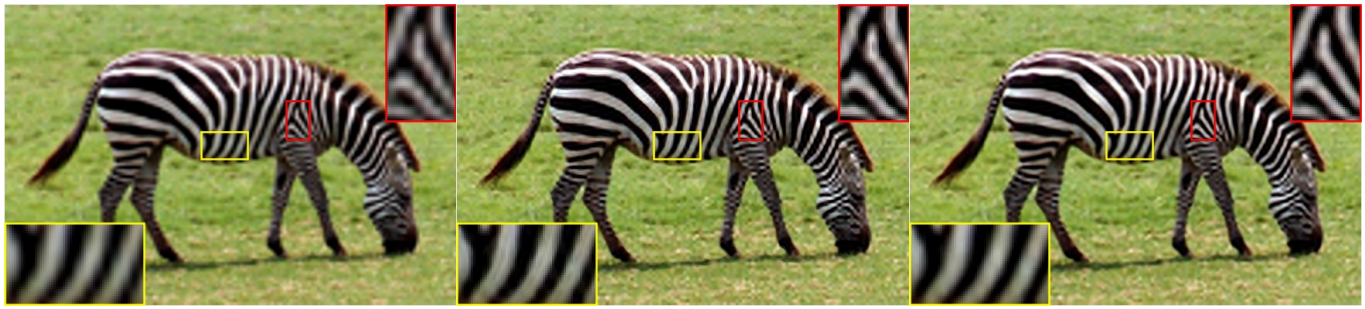
For future works, there are some proposals that we believe will improve the reconstruction quality: joining the clustering and regressor optimization in one global optimization, using different features to represent low resolution patches, and using trained dictionaries instead of sample patches to train the regressors are among them.

TABLE I
COMPARISON OF RESULTS ON DATASET SET 5

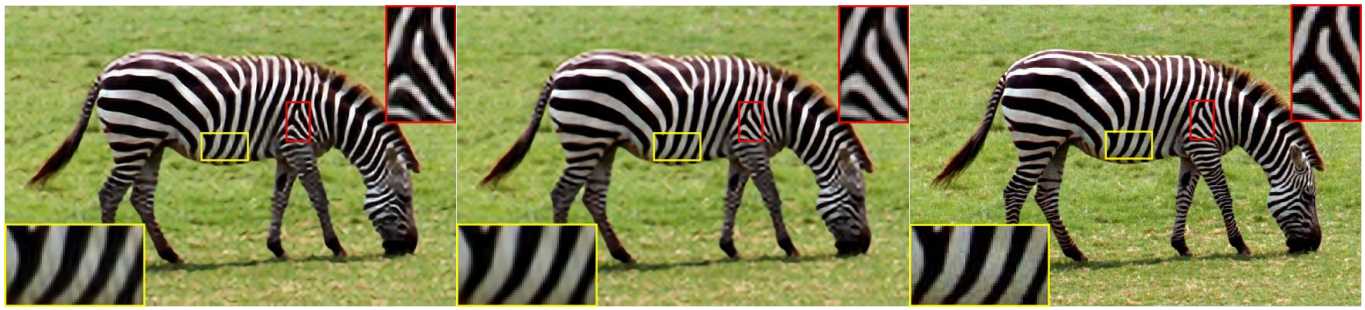
		Bicubic	SC	ANR	CNN	GELM	MELM
	PSNR (dB)	33.6972	35.9873	35.7799	36.5991	35.9938	36.5550
× 2	SSIM	0.9309	0.9508	0.9499	0.9546	0.9508	0.9540
	Time (s)	0.0018	89.0814	0.4695	3.6304	3.5966	3.9855
	PSNR (dB)	30.4114	31.8228	31.8519	32.6897	32.1538	32.5872
× 3	SSIM	0.8684	0.8953	0.8950	0.9080	0.8992	0.9075
	Time (s)	0.0010	84.3665	0.2927	3.5168	3.0982	3.1835
	PSNR (dB)	28.4407	29.5502	29.6206	30.3873	29.8957	30.2126
× 4	SSIM	0.8102	0.8375	0.8391	0.8604	0.8479	0.8589
	Time (s)	0.0013	81.9554	0.2223	3.5321	3.0772	3.1272

TABLE II
COMPARISON OF RESULTS ON DATASET SET 14

		Bicubic	SC	ANR	CNN	GELM	MELM
	PSNR (dB)	30.0205	31.6203	31.4647	32.0584	31.6311	31.9289
× 2	SSIM	0.8694	0.9022	0.9000	0.9065	0.9006	0.9043
	Time (s)	0.0020	170.9709	0.9445	9.5251	6.2078	6.2968
	PSNR (dB)	27.3418	28.3030	28.3367	28.8890	28.5087	28.7421
× 3	SSIM	0.7745	0.8094	0.8085	0.8211	0.8098	0.8170
	Time (s)	0.0017	160.4415	0.5986	8.8116	6.1859	6.2486
	PSNR (dB)	25.7979	26.5139	26.5514	27.0500	26.7305	26.9339
× 4	SSIM	0.7025	0.7331	0.7337	0.7499	0.7372	0.7463
	Time (s)	0.0016	156.4214	0.4685	9.0958	6.2102	6.2592



(a) Bicubic (b) SC (c) ANR



(d) CNN (e) MELM (f) Ground truth

Fig. 4. Visual comparison on image 'zebra' from set 14, with ×4 upscale.

(a) Bicubic (b) SC (c) ANR

(d) CNN (e) MELM (f) Ground truth

Fig. 5. Visual comparison on image 'monarch' from set 14, with $\times 4$ upscale.

ACKNOWLEDGMENT

The authors would like to thanks CAPES for financial support during their research.

REFERENCES

[1] Dinh-Hoan Trinh, M. Luong, F. Dibos, J.-m. Rocchisani, Canh-Duong Pham, and T. Q. Nguyen, "Novel Example-Based Method for Super-Resolution and Denoising of Medical Images," *IEEE Transactions on Image Processing*, vol. 23, no. 4, pp. 1882–1895, apr 2014.

[2] A. R. Pais, J. D'Souza, and R. M. Reddy, "Super-resolution video generation algorithm for surveillance applications," *The Imaging Science Journal*, vol. 62, no. 3, pp. 139–148, mar 2014.

[3] F. Li, X. Jia, and D. Fraser, "Universal HMT based super resolution for remote sensing images," in *2008 15th IEEE International Conference on Image Processing*. IEEE, 2008, pp. 333–336.

[4] H. Demirel and G. Anbarjafari, "IMAGE Resolution Enhancement by Using Discrete and Stationary Wavelet Decomposition," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1458–1460, may 2011.

[5] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel Regression for Image Processing and Reconstruction," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, feb 2007.

[6] M. Protter and M. Elad, "Super Resolution With Probabilistic Motion Estimation," *IEEE Transactions on Image Processing*, vol. 18, no. 8, pp. 1899–1904, aug 2009.

[7] R. C. Hardie, K. J. Barnard, and E. E. Armstrong, "Joint MAP Registration and High Resolution Image Estimation Using a Sequence of Undersampled Images 1 List of Figures," *Image (Rochester, N.Y.)*, vol. 12, no. 12, pp. 1621–1633, 1997.

[8] W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[9] Hong Chang, Dit-Yan Yeung, and Yimin Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. IEEE, 2004, pp. 275–282.

[10] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding," in *Procedings of the British Machine Vision Conference 2012*, no. Ml. British Machine Vision Association, 2012, pp. 135.1–135.10.

[11] R. Timofte, V. De, and L. V. Gool, "Anchored Neighborhood Regression for Fast Example-Based Super-Resolution," in *2013 IEEE International Conference on Computer Vision*. IEEE, dec 2013, pp. 1920–1927.

[12] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted Anchored Neighborhood Regression for Fast Super-Resolution," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9006, pp. 111–126.

[13] Jianchao Yang, J. Wright, T. Huang, and Yi Ma, "Image super-resolution as sparse representation of raw image patches," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2008, pp. 1–8.

[14] Jianchao Yang, J. Wright, T. S. Huang, and Yi Ma, "Image Super-Resolution Via Sparse Representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, nov 2010.

[15] R. Zeyde, M. Elad, and M. Protter, "On Single Image Scale-Up Using Sparse-Representations," 2012, vol. 1, no. 1, pp. 711–730.

[16] Jianchao Yang, Zhaowen Wang, Zhe Lin, S. Cohen, and T. Huang, "Coupled Dictionary Training for Image Super-Resolution," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3467–3478, aug 2012.

[17] 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*, vol. 38, no. 2, pp. 295–307, feb 2016.

[18] D. Liu, Z. Wang, B. Wen, J. Yang, W. Han, and T. S. Huang, "Robust Single Image Super-Resolution via Deep Networks With Sparse Prior," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3194–3207, jul 2016.

[19] J. Kim, J. K. Lee, and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 38, no. 2. IEEE, jun 2016, pp. 1646–1654.

[20] Z. Huang, L. Wang, Y. Gong, and C. Pan, "Ensemble Based Deep Networks for Image Super-Resolution," *Pattern Recognition*, mar 2017.
[21] 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*, vol. 24, no. 3, pp. 846–861, mar 2015.
[22] K. Zhang, B. Wang, W. Zuo, H. Zhang, and L. Zhang, "Joint Learning of Multiple Regressors for Single Image Super-Resolution," *IEEE Signal Processing Letters*, vol. 23, no. 1, pp. 102–106, jan 2016.
[23] Y. Romano, J. Isidoro, and P. Milanfar, "RAISR: Rapid and Accurate Image Super Resolution," *IEEE Transactions on Computational Imaging*, vol. 1606.01299, pp. 1–1, jun 2017.
[24] L. An and B. Bhanu, "Image super-resolution by extreme learning machine," in *2012 19th IEEE International Conference on Image Processing*, vol. 1, no. 1. IEEE, sep 2012, pp. 2209–2212.
[25] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2. IEEE, 2004, pp. 985–990.
[26] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, dec 2006.
[27] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, apr 2012.
[28] P. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, mar 1998.
[29] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, jul 2006.
[30] H. Farid and E. Simoncelli, "Differentiation of Discrete Multidimensional Signals," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 496–508, apr 2004.
[31] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, apr 2004.