DEEP NETWORK FOR IMAGE SUPER-RESOLUTION WITH A DICTIONARY LEARNING LAYER

Yang Liu ⋆ *Qingchao Chen* † *Ian Wassell* ⋆

⋆ Computer Laboratory, University of Cambridge, Cambridge, UK
†Department of Electronic and Electrical Engineering, University College London, London, UK

ABSTRACT

The aim of single image super-resolution (SR) is to generate a high-resolution (HR) image from a low-resolution (LR) observable image. In this paper, we address this task by integrating sparse coding and dictionary learning schemes into an end-to-end deep architecture. More specifically, we propose a new non-linear dictionary learning layer composed of a finite number of recurrent units to solve the sparse codes and also to yield the relevant gradients to update the dictionary. In addition, we present a new deep network architecture using the proposed non-linear layers, where two separate parallel dictionaries are adopted to represent the LR and HR images respectively. The whole network is optimized by back propagation, constraining not only reconstruction errors between the restored and the ground truth HR images but also between the sparse codes of the LR and HR image pairs. Various datasets are used to evaluate the performance of the proposed approach and it is shown to outperform many state-of-the-art single image super-resolution algorithms.

Index Terms— Super-resolution, Dictionary Learning, Deep learning

1. INTRODUCTION

Single image super-resolution (SR) is of great importance in video applications aims to find a mapping from the low-resolution (LR) observations to yield a high-resolution (HR) image. With the ill-posed nature of this task and the severe loss of information between the LR and HR images, numerous algorithms have been proposed, but they are far from being satisfactory for many practical applications [1][2].

For the SR solution, the conventional approach is to regard the task as an ill-posed inverse problem, estimating the degradation matrix, using numerous priors related to the image characteristics. However, due to the scarcity of this information, the solution is still under-determined. Better SR performance can be delivered by example-based methods, that represent an HR patch as a sparse linear combination of dictionary atoms learned from an external database or from similar patches from the LR image itself (i.e., self-similar patches) [3]. In recent years, with the growing popularity of the deep learning architecture, neural networks with a deep architecture have achieved significant improvements for image SR. Multiple layers of collaborative auto-encoders are stacked together in [3] for robust matching of self-similar patches. Deep convolutional neural networks (CNN) [4] and deconvolutional networks [5] have been designed that directly learn the non-linear mapping from LR space to HR space. The networks in [3] [4] are built with generic architectures, which means all their knowledge about SR is learned from training data. However, some studies have argued based on the results of experiments, that although improvements in the end-to-end deep architectures have yielded performance gains, domain expertise cannot be ignored even when employing deep architectures. They further argued that a combination of sparse prior knowledge [6] with the end-to-end architecture would generate more robust and efficient SR methods [2]. However, the authors in [2] have only considered the reconstruction error between the HR images and the generated ones, but not between the sparse codes of the LR and HR representations. Furthermore, they followed the learned iterative shrinkage and thresholding algorithm (LISTA) [7] [8] framework, consequently, the sparse coding and dictionary updating steps are still implemented as two separate steps/layers in the network architecture, that cannot efficiently update the dictionary.

In this paper, instead of using the LISTA algorithm for directly solving the sparse coefficients offline, to our best knowledge, we are the first to design a new non-linear layer composed of a finite number of small recurrent units to integrate the sparse coding and dictionary learning framework into the end-to-end deep architecture. More specifically, this newly proposed dictionary learning layer (DLL) can solve the sparse coefficients while also contributing the gradient flow to update the dictionary in each recurrent iteration. To address the SR task, we build a new network architecture called the Dictionary Learning Network (DLN), that includes two DLL layers to represent LR and HR images in parallel, that exhibits an enhanced capacity to capture the local detailed information. In the following, we will first introduce and formulate our model in detail in Section 2. In Section 3, the implementation details are provided and our DLN is validated using extensive datasets. Finally, we compare our methods with others and conclude our paper in Section 4.

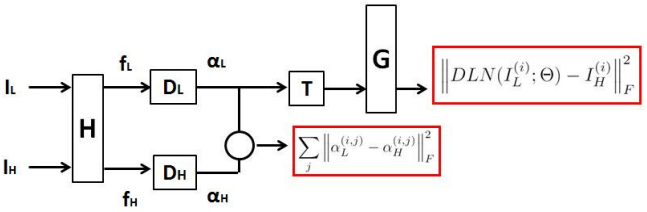

Fig. 1. Overall dictionary learning based network architecture. There are 5 trainable layers; two convolutional layers $\boldsymbol{H}$ and $\boldsymbol{G}$, two non-linear dictionary learning layers designed for LR and HR patches, that are parametrized as $\boldsymbol{D}_L$ and $\boldsymbol{D}_H$ respectively and one linear layer $\boldsymbol{T}$ to reconstruct the HR image patch that is parametrized with $\boldsymbol{D}_H$.

2. DICTIONARY LEARNING NETWORK

2.1. Problem Formulation

Similar to most patch-based deep learning SR methods, our dictionary learning based network aims to learn the transformation from the bicubic-upscaled LR image $\boldsymbol{I}_L$ to a corresponding HR image $\boldsymbol{I}_H$ in an end-to-end manner. Fig.1 shows the main network structure, and the details of each layer are discussed as follows.

To extract the feature of each patch, the input LR image $\boldsymbol{I}_L$ and HR image $\boldsymbol{I}_H$ first go through a convolution layer H, composed of m filters with a spatial size of $h \times h$. In this way, the dimension of each input image patch is $h \times h$ and the the output feature of each LR or HR patch is represented as $\boldsymbol{f}_L \in \mathbb{R}^m$ or $\boldsymbol{f}_H \in \mathbb{R}^m$ respectively. The feature representations $\boldsymbol{f}_L$ and $\boldsymbol{f}_H$ are then sent to two non-linear dictionary learning layers (DLL), $\boldsymbol{D}_L$ and $\boldsymbol{D}_H$ respectively, each of which contains a finite number of recurrent units to mimic the sparse coding procedure.

To extract the local discriminative properties among LR and HR patches, a coupled dictionary pair $\boldsymbol{D}_L \in \mathbb{R}^{m\times n}$ and $\boldsymbol{D}_H \in \mathbb{R}^{m\times n}$ are designed within the deep architecture for the LR and HR feature space respectively. In addition, we constrain the corresponding sparse coding vectors $\boldsymbol{\alpha}_L \in \mathbb{R}^n$ and $\boldsymbol{\alpha}_H \in \mathbb{R}^n$ such that they are similar in the two latent spaces.

Specifically, a pair of coupled dictionaries $\boldsymbol{D}_L$ and $\boldsymbol{D}_H$ is ideally learned if the following equations are satisfied for any corresponding LR and HR image patches.

$$\begin{aligned}
&\min_{\boldsymbol{\Theta}} \sum_i \left\|\boldsymbol{\alpha}_L^{(i)}-\boldsymbol{\alpha}_H^{(i)}\right\|_F^2 \\
&s.t.\ \boldsymbol{\alpha}_L^{(i)} = arg\min_{\boldsymbol{\alpha}_L^{(i)}} \left\|\boldsymbol{f}_L^{(i)}-\boldsymbol{D}_L\boldsymbol{\alpha}_L^{(i)}\right\|_F^2 + \lambda_1\left\|\boldsymbol{\alpha}_L^{(i)}\right\|_1 \quad (1)\\
&\boldsymbol{\alpha}_H^{(i)} = arg\min_{\boldsymbol{\alpha}_H^{(i)}} \left\|\boldsymbol{f}_H^{(i)}-\boldsymbol{D}_H\boldsymbol{\alpha}_H^{(i)}\right\|_F^2 + \lambda_2\left\|\boldsymbol{\alpha}_H^{(i)}\right\|_1
\end{aligned}$$

In general, the parameters within the DLL can be trained in an end-to-end manner through standard back propagation. Once the dictionaries in the DLL are learned, the corresponding sparse coding vectors for each patch $\boldsymbol{\alpha}_L^{(i)} \in \mathbb{R}^n$ or $\boldsymbol{\alpha}_H^{(i)} \in \mathbb{R}^n$ can be estimated efficiently by feeding $\boldsymbol{f}_L^{(i)}$ or $\boldsymbol{f}_H^{(i)}$ through the corresponding non-linear DLL. The detailed description of the design of DLL and the optimisation rules are discussed in 2.2.

The sparse coding $\boldsymbol{\alpha}_L$ is then multiplied with the HR dictionary $\boldsymbol{D}_H \in \mathbb{R}^{m\times n}$ in the next linear layer T, and the size of the reconstructed HR patch is $h \times h = m$.

To return the reconstructed HR patches into the corresponding positions of the HR image $\boldsymbol{I}_y$, a convolution layer G is applied composed of m convolution filters with a spatial size of $g \times g$. The filter size g is determined by the number of neighboring patches that contribute to the same pixel in the HR image $\boldsymbol{I}_y$. These convolution filters aim to assign different weights to patches and aggregate the weighted average to give the final HR image prediction $\boldsymbol{I}_y$.

To sum up, five trainable layers are designed in our DLN network as shown in Fig.1. It contains two convolution layers $\boldsymbol{H}$ and $\boldsymbol{G}$, two non-linear DLL for LR and HR feature spaces respectively and a linear layer $\boldsymbol{T}$ to reconstruct the final HR image patch. It is worth noting that the dictionary $\boldsymbol{D}_H$ used in the non-linear and linear layers need to share the same weights during both the training and testing stages.

To train the designed network, the mean square error (MSE) criterion is adopted as the cost function and the formalized overall optimisation objective function can be expressed as follows:

$$\min_{\boldsymbol{\Theta}} \sum_i \{\left\|DLN(\boldsymbol{I}_L^{(i)};\boldsymbol{\Theta})-\boldsymbol{I}_H^{(i)}\right\|_F^2 + \lambda\sum_j \left\|\boldsymbol{\alpha}_L^{(i,j)}-\boldsymbol{\alpha}_H^{(i,j)}\right\|_F^2\}, \quad (2)$$

where $\boldsymbol{I}_L^{(i)}$ and $\boldsymbol{I}_H^{(i)}$ are the i-th pair of LR/HR training data. In addition, note that $\boldsymbol{\alpha}_L^{(i,j)}$ and $\boldsymbol{\alpha}_H^{(i,j)}$ represent the sparse codes of the j-th patch from the i-th image pair under the dictionaries $\boldsymbol{D}_L$ and $\boldsymbol{D}_H$ respectively. Furthermore, the term $DLN(\boldsymbol{I}_L^{(i)})$ represents the synthesized HR image transformed from the LR image $\boldsymbol{I}_L$, using the DLN model parametrized by the set $\boldsymbol{\Theta}$. Specifically in Eq.(2), the first term is the image level reconstruction error and the second term represents the similarity error of the HR and LR sparse codes. The parameter λ is used to balance the contribution of the two terms and all the parameters in the network can be trained using the standard back-propagation algorithm in an end-to-end manner. Our experiments in section 3 verify that minimizing the objective function (2) is able to improve the performance of the single image SR task.

2.2. Non-linear Dictionary Learning Layer

In this section, we introduce the details of the fast yet accurate non-linear DLL. In general, the DLL is able to compute

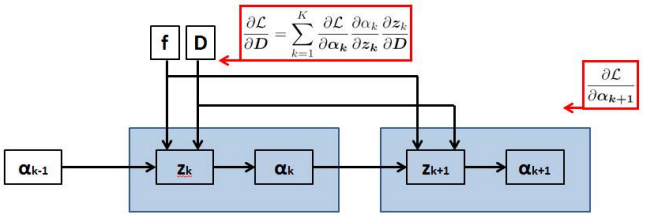

Fig. 2. The design of a non-linear dictionary learning layer. Each blue box in the figure represents a recurrent unit, which corresponds to one iteration in the sparse coding procedure. The dictionary among different recurrent units are shared within a non-linear dictionary learning layer.

the corresponding sparse codes efficiently given the feature representation. The design of this layer is inspired by the iterative shrinkage and thresholding algorithm (ISTA) and the on-line dictionary learning method [9]. This non-linear dictionary learning layer contains a finite number of recurrent units, each of which exhibits a similar function to that of an iteration in the ISTA algorithm. More specifically, we build this layer as a data flow graph, where a node represents a variable and the data flow between two variables represents the directed edge. Given a feature representation, it flows over the graph and generates its corresponding sparse coding vector. In contrast to LISTA [8], that only updates the dictionary once after the sparse coding procedure ends, we use a shared dictionary as a parameter in each recurrent unit and any intermediate sparse coding vector output within the ISTA procedure can contribute to update the dictionary $\boldsymbol{D}$. As shown in the blue boxes in Fig.2, the operation between different nodes in the data flow graph can be represented as:

$$\boldsymbol{z}_{k+1} = \boldsymbol{\alpha}_k - \gamma \boldsymbol{D}^T(\boldsymbol{f} - \boldsymbol{D}\boldsymbol{\alpha}_k) \tag{3}$$

$$\boldsymbol{\alpha}_{k+1} = max(|\boldsymbol{z}_{k+1}| - \boldsymbol{t}, 0)\frac{\boldsymbol{z}_{k+1}}{|\boldsymbol{z}_{k+1}|}, \tag{4}$$

where γ represents the step size and $\boldsymbol{t}$ represents the thresholding vectors. Since these two parameters can be learned in LISTA, which fits well to the existing data set, the sparse coding steps can be achieved in only a few iterations, where we denote the number of such iterations within ISTA procedure as K. As the overall objective function in (2) does not depend on $\boldsymbol{D}$ explicitly, it is difficult to compute the gradient with respect to $\boldsymbol{D}$ in each recurrent unit. Therefore, we consider the dictionary $\boldsymbol{D}$ as a special parameter implicitly in each recurrent unit and propose to compute the gradient of the loss function L with respect to $\boldsymbol{D}$ using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_k} \frac{\partial \boldsymbol{\alpha}_k}{\partial \boldsymbol{z}_k} \frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{D}}, \tag{5}$$

in which

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_k} = \begin{cases} \prod_{k+1}^{K} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_k} \frac{\partial \boldsymbol{\alpha}_k}{\partial \boldsymbol{z}_k} \frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{\alpha}_{k-1}} & if\ k < K \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}_k} & if\ k = K \end{cases} \tag{6}$$

$$\frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{D}} = \gamma(\boldsymbol{f} - 2\boldsymbol{D}\boldsymbol{\alpha}_{k-1}) \tag{7}$$

Each element in $\frac{\partial \boldsymbol{\alpha}_k}{\partial \boldsymbol{z}_k}$ is set to 0 if the q^{th} element $\boldsymbol{\alpha}_k^q = 0$, otherwise, it is set to 1.

In fact, the non-linear DLL is designed based on our domain knowledge in sparse coding. It is worth noting that in the overall DLN architecture, the two non-linear DLLs are controlled with different dictionaries, namely $\boldsymbol{D}_L$ and $\boldsymbol{D}_H$ to represent the LR and HR samples in parallel. Until now all the parameters within the overall DLN can be trained through standard back-propagation. We will see in the experiments that this new layer and the overall DLN structure can generate better SR results, and can be trained more rapidly and with the use of fewer parameters than can the conventional CNN structure.

3. EXPERIMENTS

3.1. Implementation Details

In the following experiments, the proposed model is trained using the Theano and Lasagne package [10] on a PC with an Intel 3.07GHz CPU and a TITAN-X GPU. We determine the number of nodes in each layer of our DLN mainly according to the corresponding settings used in sparse coding [11] and sparse coding network (SCN) in [2]. The input patch size of the low-resolution image is set to $h = 9$ and the feature dimension of each patch and the dictionary size are set as m=100, n=128 respectively. For the output high-resolution image, the patch size and the patch aggregation filter size $g = 5$. In addition, all the convolution layers have a stride of 1. To ensure correct reconstruction, the same mean and variance are used in both the low-resolution patch $\boldsymbol{f}_L$ and the high-resolution patch $\boldsymbol{f}_H$.

For the parameter initialization, we use uniform weights to initialize layers $\boldsymbol{H}$ and $\boldsymbol{G}$. For the two non-linear dictionary learning layers, we randomly set $\boldsymbol{D}_L$ and $\boldsymbol{D}_H$with Gaussian noise entries. The standard stochastic gradient descent algorithm is adopted to train the proposed networks with a mini-batch size of 64. The learning rate is set as 0.001 with a momentum of 0.9.

In testing, we follow the objective evaluations in [4] to ensure fair comparisons. Specifically, we employed the same method to shave the image border, and in addition, we extended the boundary of the original image by using mirror reflections of itself and then cropped input samples having overlap. Note that we also adopted the scheme commonly used for comparison, whereby our DLN is only utilized in the luminance channel and bicubic interpolation is applied to the

Table 1. Comparison of PSNR (SSIM) performance (PSNR in dB) on three test data sets for different methods. The bold entry in each column indicates the best performance. The performance gain of our model over the best of all the others is shown in the last row.

DataSet	Set 5			Set 14			BSD 100		
Upscaling Factor	×2	×3	×4	×2	×3	×4	×2	×3	×4
A+	36.55 (0.9544)	32.59 (0.9088)	30.29 (0.8603)	32.28 (0.9056)	29.13 (0.8188)	27.33 (0.7491)	30.78 (0.8773)	28.18 (0.7808)	26.77 (0.7085)
CNN	36.34 (0.9521)	32.39 (0.9033)	30.09 (0.8530)	32.18 (0.9039)	29.00 (0.8145)	27.20 (0.7413)	31.11 (0.8835)	28.20 (0.7794)	26.70 (0.7018)
CNN-L	36.66 (0.9542)	32.75 (0.9090)	30.49 (0.8628)	32.45 (0.9067)	29.30 (0.8215)	27.50 (0.7513)	31.36 (0.8879)	28.41 (0.7863)	26.90 (0.7103)
SCN	36.93 (0.9552)	33.10 (0.9144)	30.86 (0.8732)	32.56 (0.9074)	29.41 (0.8238)	27.64 (0.7578)	31.40 (0.8884)	28.50 (0.7885)	27.03 (0.7161)
DLN	**37.68 (0.9629)**	**33.85 (0.9267)**	**31.51 (0.8879)**	**32.98 (0.9122)**	**29.86 (0.8314)**	**27.98 (0.7679)**	**32.21 (0.8967)**	**28.97 (0.7983)**	**27.31 (0.7269)**
Improvement	0.75 (0.0077)	0.75 (0.0123)	0.65 (0.0147)	0.42 (0.0048)	0.45 (0.0076)	0.34 (0.0101)	0.81 (0.0083)	0.47 (0.0098)	0.28 (0.0108)

chrominance channels. We also implemented the cascaded DLN in a similar way to that in [2]. In order to ensure that the transformed low-resolution image is at least as large as the desired dimension, an image is upscaled by 2 times repeatedly. Then a bicubic interpolation is used to downscale it to the target resolution if necessary.

3.2. Experiment and Results Analysis

For evaluating the performance of our model and to enable comparison with other approaches, the same data and protocols as used in [12] are employed (that are commonly adopted in SR literature). In general, our DLN model is trained based on 91 samples and tested on Set5 (5 images)[13], Set14(14 images) [14] and BSD100 (100 images) [15] with different scaling factors. Note that the original high-resolution images are downsized and bicubic interpolated to generate LR-HR training sample pairs and test samples. The training data are augmented with translation, rotation and scaling.

We compare the proposed DLN model with other recent SR methods on all the images in Set5, Set14 and BSD100 for different upscaling factors. It can be seen from Table 1 that the PSNR and structural similarity (SSIM) [16] are measured to evaluate all the methods, specifically, adjusted anchored neighborhood regression (A+) [17], CNN [4], CNN trained with a larger model size and more data (CNN-L) [18], the sparse coding network (SCN) [2] and the proposed DLN. As shown in Table 1, our DLN model performs consistently better than all the other methods in terms of both PSNR and SSIM. Although SCN improves upon CNN-L and CNN using a smaller dataset, it is still not as good as DLN trained with the same dataset. The reason for this is two-fold: first, DLN utilizes two separate dictionaries to represent the low and high-resolution images respectively, which exhibits a better capacity to capture the locally detailed differences and the discriminative properties between image pairs. In addition, it ensures the reconstruction of global properties of image pairs by combining constraints concerning image level reconstruction error and that of achieving similar sparse codes from the two dictionaries. Second, DLN has an advantage over SCN owing to the new structure proposed for sparse coding and the dictionary learning. In the experiment, different numbers of recurrent units K have been evaluated for DLN, and we find increasing K from 2 to 10 only improves performance by less than 0.1dB. Consequently, we usually only need 2 recurrent units to achieve the sparse coding vector in the forward path. To sum up, as described in section 2, the better design of the non-linear layers, (having a finite number of cascaded recurrent units) is able to provide not only better sensitivity to capture the discriminative properties between image pairs but also ensures a better convergence rate.

4. CONCLUSION

This paper integrates knowledge of the sparse coding and dictionary learning into end-to-end deep architectures for the SR task and shows state of the art performance via extensive experiments on various SR datasets. Furthermore, composed of a series of recurrent units, our newly proposed nonlinear DLL is implemented effectively and efficiently, and exhibits superior performance in solving the sparse code and updating the dictionary. The performance improvements shown in this paper also arise from our network architecture design, where two separate dictionaries are learned in parallel by two DLL layers to capture the properties of the LR and HR images respectively. Future work will aim at providing a theoretical analysis of the DLL layer.

5. REFERENCES

[1] Zhouchen Lin and Heung-Yeung Shum, "Fundamental limits of reconstruction-based superresolution algorithms under local translation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 83–97, 2004.

[2] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang, "Deep networks for image super-resolution with sparse prior," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 370–378.

[3] Zhen Cui, Hong Chang, Shiguang Shan, Bineng Zhong, and Xilin Chen, "Deep network cascade for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 49–64.

[4] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 184–199.

[5] Christian Osendorfer, Hubert Soyer, and Patrick Van Der Smagt, "Image super-resolution with fast approximate convolutional sparse coding," in *International Conference on Neural Information Processing*. Springer, 2014, pp. 250–257.

[6] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma, "Image super-resolution via sparse representation," *IEEE transactions on image processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[7] Christopher J Rozell, Don H Johnson, Richard G Baraniuk, and Bruno A Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural computation*, vol. 20, no. 10, pp. 2526–2563, 2008.

[8] Karol Gregor and Yann LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 399–406.

[9] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 689–696.

[10] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2016.

[11] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas Huang, "Coupled dictionary training for image super-resolution," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[12] Radu Timofte, Vincent De Smet, and Luc Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1920–1927.

[13] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[14] Kaibing Zhang, Xinbo Gao, Dacheng Tao, and Xuelong Li, "Single image super-resolution with non-local means and steering kernel regression," *IEEE Transactions on Image Processing*, vol. 21, no. 11, pp. 4544–4556, 2012.

[15] 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 *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. IEEE, 2001, vol. 2, pp. 416–423.

[16] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[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*. Springer, 2014, pp. 111–126.

[18] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016.