This CVPR2015 workshop paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Self-Tuned Deep Super Resolution

Zhangyang Wang†, Yingzhen Yang†, Zhaowen Wang‡, Shiyu Chang†,
Wei Han†, Jianchao Yang♢, and Thomas Huang†
†Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
‡Adobe Systems Inc, San Jose, CA 95110, USA
♢ Snapchat Inc, Venice, CA 90291, USA
{zwang119, yyang58, wang308, chang87, weihan3, t-huang1}@illinois.edu
zhanwang@adobe.com jianchao.yang@snapchat.com

Abstract

Deep learning has been successfully applied to image super resolution (SR). In this paper, we propose a deep joint super resolution (DJSR) model to exploit both external and self similarities for SR. A Stacked Denoising Convolutional Auto Encoder (SDCAE) is first pre-trained on external examples with proper data augmentations. It is then fine-tuned with multi-scale self examples from each input, where the reliability of self examples is explicitly taken into account. We also enhance the model performance by sub-model training and selection. The DJSR model is extensively evaluated and compared with state-of-the-arts, and show noticeable performance improvements both quantitatively and perceptually on a wide range of images.

1. Introduction

Super-resolution (SR) algorithms aim to construct a high-resolution (HR) image from one or multiple low-resolution (LR) inputs. Being ill-posed, SR has to resort to strong image priors, ranging from the simplest analytical smoothness assumptions, to more complicated statistical and structural priors [25], [7]. The most popular SR methods rely on a large and representative external set of image pairs to learn the mapping between LR and HR image patches [26]. Those methods are known for their capabilities to produce plausible image appearances. However, there is no guarantee that an arbitrary input patch can be well matched or represented by a pre-chosen external set. When there are rarely matching features for the input, external examples are prone to produce either noise or over-smoothness [23]. Meanwhile, image patches tend to recur within the same image [8], [23], or across different image scales [7]. The self similarity property provides self examples that are highly relevant to the input, but only of a limited number. Due to the insufficiency of self examples, their mismatches often result in more visual artifacts. It is recognized that external and self example-based SR methods each suffer from their inherent drawbacks [20].

The joint utilization of both external and self examples has been first studied for image denoising [28]. Mosseri et. al. [14] proposed that image patches have different preferences towards either external or self examples for denoising. Such a preference is in essence the tradeoff between noise-fitting versus signal-fitting. Burger et. al. [1] proposed a learning-based approach that automatically combines denoising results from an self example and an external example-based method. In SR literature, the authors in [5] incorporated both a local autoregressive (AR) model and a nonlocal self similarity regularization term, into the sparse representation framework. Yang et. al. [24] learned the approximated nonlinear SR mapping function from external examples with the help of in-place self similarity. More recently, a joint SR model was proposed in [19], [20] [21], to adaptively combine the advantages of both external and self examples. It is observed in [20] that external examples contribute to visually pleasant SR results for relatively smooth regions, while self examples reproduce recurring singular features of the input. The complementary behavior has been similarly verified in the the image denoising literature [1].

More recently, inspired by the great success achieved by deep learning (DL) models in other computer vision tasks [10], there is a growing interest in applying deep architectures to image SR. A Super-Resolution Convolutional Neural Network (SRCNN) was proposed in [4]. Thanks to the end-to-end training and the large learning capacity of the CNN, the SRCNN obtains significant improvements over classical non-DL methods. In SRCNN, the information exploited for reconstruction is comparatively larger than that used in previous sparse coding approaches. However, the SRCNN has not taken any self similarity property into account. The authors in [2] proposed the deep network cas-

cade (DNC) to embed self example-based approach to auto-encoders (AEs). In each layer of the network, patchwise non-local self similarity search is first performed to enhance high-frequency details, and thus the whole model is not specifically designed to be an end-to-end solution. So far, there lacks a principled approach to utilize self similarity to regularize deep learning models, not only for SR, but also for general image restoration applications.

In this paper, we propose a unified deep learning framework, to joint utilize both the wealth of external examples, and the power of self examples specifically to the input. We name our proposed model *deep joint super resolution* (**DJSR**). While the mutually reinforcing properties of external and self similarities are utilized in classical example-based methods [5], [24], [20], to our best knowledge, DJSR is the first to adapt deep models for joint similarities. The major contributions are summarized as multi-folds:

- We pre-train the model using an external set with data augmentations. We then fine-tune it using self-example pairs from the input image. Such a framework can be easily extended to other applications.
- We propose to sample a large pool of self-example pairs using multi-scale self similarity, each of which is assigned a confidence weight during training. That alleviates the insufficiency of reliable self examples.
- We extend DJSR into several dedicated sub-models, and conduct selective training and patch processing.

Connecting SR to Domain Adaption The idea of DJSR has certain connections to domain adaption or transfer learning [9]. For domain adaption, given a source domain having sufficient labeled data for training, and a target domain with insufficient labeled data and a different distribution, the problem is to have the model trained on the source domain generalize well on the target domain. In our setting, LR-HR pairs resemble the data-label tuples. The DJSR model is first learned on the source domain of external examples, and then adapted to the target domain of self samples from the testing image. That explains why DJSR could outperform previous models based on either external examples (applying source domain models directly to the target domain) or self examples (relying on target domain only to train models) from a domain adaption perspective.

2. Pre-Training Using External Examples

Several deep architectures have been explored for SR previously. The authors of DNC [2] referred to a collaborative local auto-encoder (CLA) to be stacked to form a cascade. However, auto-encoders (AEs) rely mostly on fully-connected models and ignore the 2D image structure [17]. In SRCNN [4], a fully convolutional network is learned to predict the nonlinear LR-HR mapping. Such a model has a clear analogy to classical sparse coding methods [25].

In [12], a Convolutional Auto Encoder (CAE) was proposed to learn features using a hierarchical unsupervised feature extractor while preserving spatial locality. CAEs can be stacked to form a Stacked Convolutional Auto Encoder (SCAE), where each layer receives its input from a latent representation of the layer below. It was further revealed in [17] that auto encoders are prone to learn trivial projections and the learned filters are usually subject to random corruptions. Denoising was thus suggested as a training criterion [17] to learn robust structural features, which also proves to be effective for image restoration tasks besides the original classification setting [22]. We employ a Stacked Denoising Convolutional Auto-Encoder (SDCAE) [12] to reconstruct HR images from its stochastically corrupted LR versions. Such an architecture combines the intuitive idea of AEs, and the power of CNNs to capture 2-D structures efficiently. Note that more potential alternatives, such as SRCNN, can possibly be adapted here.

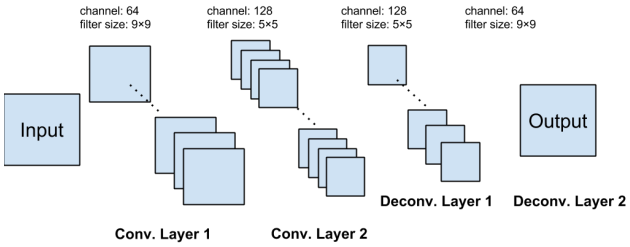

Figure 1. The SDCAE architecture for SR.

While there are multiple SCAE implementations available, we adopt the implementation by [15] [1] as it has shown some improvements over the one in [12] on the CIFAR-10 benchmark. Their model is similar to the network in [27] but without using sparse coding, and introducing zero-biases [13] and ReLUs in the convolutional layers. To convert the SCAE into a SDCAE, all we need to do is to add a stochastic corruption (we use additive isotropic Gaussian noise) step operating on the input. Assuming that the original images are downsized by a scale s to generate LR-HR example pairs for both training and evaluation, the SDCAE architecture is depicted in Fig. 1, where the input is a LR image and the output is its HR counterpart. The trained network fits SR with a factor of s.

3. Fine-Tuning Using Self Examples

3.1. Self-example pairs by Multi-Scale Similarity

In [2], the authors enhanced high-frequency by employing non-local self similarity (NLSS) search over the succes-

[1] https://github.com/ifp-uiuc/anna

Algorithm 1 Generate A Hierarchy of Self Examples Associated with Confidence Weights

Require: Input image $\mathbf{Y}$, scaling factor s, number of scales N, number of self-example pairs per patch m.

1: Upsample $\mathbf{Y}$ for $\frac{N}{2}-1$ times with a factor of s.
2: Downsample $\mathbf{Y}$ for $\frac{N}{2}-1$ times with a factor of s.
3: For each patch in original $\mathbf{Y}$, find all spatially co-located patches in other $N-1$ resized versions.
4: For each co-located patch, find $\frac{m}{(N-1)}$ best matched examples from its immediate upscaled version, using the method in (1).
5: Make the co-located patch and each of its matched example a self-example pair.
6: Record the NN matching error for each match.
7: Take the negative exponents of all matching errors, and normalize them between [0,1]. Those will be used as the associated confidence weights.

Ensure: m self-example pairs per patch in $\mathbf{Y}$ with weights.

sive blurred and downscaled versions of the input image. By combining those internal matches, the estimated patch usually contains more abundant texture information. However, it overlooks the across-scale similarity properties of natural images [7], [24], that singular features like edges and corners in small patches tend to repeat almost identically in their slightly upscaled versions. In addition, such a pre-processing step is not jointly optimized with the deep network cascade.

Freedman and Fattal [7] applied the "high frequency transfer" method to search for the high-frequency component of a target HR patch, by NN patch matching across scales. Let $\mathbf{X}$ denote the HR image to be estimated from the LR input $\mathbf{Y}$. $\mathbf{X}_{ij}$ and $\mathbf{Y}_{ij}$ stand for the (i,j)-th ($i,j = 1,2...$) patch from $\mathbf{X}$ and $\mathbf{Y}$, respectively. Defining a linear interpolation operator $\mathcal{U}$ and a downsampling operator $\mathcal{D}$, for the input LR image $\mathbf{Y}$, we first obtain its initial upsampled image $\mathbf{X}' = \mathcal{U}(\mathbf{Y})$, and a smoothed input image $\mathbf{Y}' = \mathcal{D}(\mathcal{U}(\mathbf{Y}))$. Given the smoothed patch $\mathbf{X}'_{ij}$, the missing high-frequency band of each unknown patch $\mathbf{X}_{ij}$ is predicted by first solving a NN matching (1):

$$(m,n) = \arg\min_{(m,n)\in\mathcal{W}_{ij}} \|\mathbf{Y}'_{mn} - \mathbf{X}'_{ij}\|_F^2, \qquad (1)$$

where $\mathcal{W}_{ij}$ is defined as a searching window on image $\mathbf{Y}'$. With the co-located patch $\mathbf{Y}_{mn}$ from $\mathbf{Y}$, the high-frequency band $\mathbf{Y}_{mn} - \mathbf{Y}'_{mn}$ is pasted onto $\mathbf{X}'_{ij}$, i.e., $\mathbf{X}_{ij} = \mathbf{X}'_{ij} + \mathbf{Y}_{mn} - \mathbf{Y}'_{mn}$.

Note the above methodology could be applied to construct self-example pairs $\{\mathbf{Y}_{ij}, \mathbf{X}_{ij}\}$ for a input image $\mathbf{Y}$. It is thus straightforward to consider adopting those self-example pairs to fine-tune our pre-trained network. However, two problems obstacle such a practice:

- **Insufficiency of Informative Examples** The amount of self examples is usually far less than the size of external training sets. For example, a 256×256 input image can generate at most $(256-9+1)^2$= 61,504 patches of a small size 9×9 (and thus the same amount of self-example pairs) and a minimum stride of 1 [6]. Moreover, the information of those example pairs is also far less rich.

- **Limited Reliability** In essence, a part of input patches may be identified with few discernible repeating patterns. They might thus not be able to find good matches $\mathbf{Y}'_{mn}$ within the same image, which constitutes the visual artifacts in previous high frequency transfer methods [7]. Besides, The matching of $\mathbf{X}'_{ij}$ over $\mathbf{Y}'$ makes the core step of the high frequency transfer scheme. However, NN matching (1) is not reliable under noise and outliers in LR images.

To resolve the above raised concerns, we sample a hierarchy of self examples from multiple scales, each of which is associated with a confidence weight calculated by the NN matching error from (1). The key idea is to exploit cross-scale patch redundancy embedded between multiple neighborhood scales. The steps are outlined in Algorithm I.

3.2. Weighted Back Propagation

The self-example pairs obtained from Algorithm 1 can be used to fine-tune the pre-trained SDCAE, making it specially adapted for the input. To incorporate the reliability of the self-example pairs into the process, a variant of standard back propagation, called *Weighted Back Propagation* (**WBP**), is developed to alleviate the negative impacts of bad examples, without sacrificing the benefits of abundant training data. In particular, assuming that ω is the normalized confidence weight for the current self-example pair, let η_f denote the learning rate for fine tuning and δ the gradient, the weight matrices W_i (i is the layer index) are updated as:

$$\delta_{i+1} = \omega \cdot (\eta_f \cdot \frac{\partial L}{\partial W_i} + 0.9 \cdot \delta_i), \quad W_{i+1} = W_i + \delta_{i+1} \qquad (2)$$

Note each example pair possesses a different (and precalculated) ω. Such an importance weighting strategy has been commonly applied to transfer learning problems [11]. Yet to our best knowledge, there is no similar work in deep learning. In all experiments, the learning rate η_f for fine-tuning is set to be 0.5 by default, and will not be annealed. That large value is empirically found to work well, leading to a better presence of self similarity in final SR results.

4. Sub-model Training and Selection

Previous DL-based image SR methods aim at learning one model that is capable to represent various image structures. Such a model lacks the adaptivity to local struc-

tures. In some cases, It might also lead to a model of overly high complexity and redundancy [5]. When learning regressors from LR to HR patches, it is observed that regressors have different specialities at dealing with certain patches [3]. Following this idea, external examples are partitioned into many clusters, each of which consists of patches with similar patterns and can be used to pre-train a sub SDCAE model. Next, for each input patch, the most relevant sub-model is first selected and then fine-tuned by its own self-example pairs. Since a given patch can be better represented by the adaptively selected sub-models, the whole HR image can be more accurately reconstructed.

Provided with an external set, we first use the high-pass filtering output of each LR patch as the feature for clustering. It allows us to focus on the edges and structures of image patches. We then adopt K-means algorithm to partition the whole set into K clusters, where μ_i denotes the centroid of i-th cluster, $i = 1, 2, ..., K$. During model fine-tuning (and testing), the best sub-model is chosen based on the minimum distance between the LR patch and the centroids. As suggested by [5], let $U = [\mu_1, \mu_2, ...\mu_K]$, its PCA transformation matrix is obtained by applying SVD to the co-variance matrix of U. We can compute the distance between input patch and cluster centroids more robustly in the subspace spanned by the most significant eigenvectors [5].

5. Experiments

5.1. Implementation

The SDCAE is learned from an external training set with 91 images [26], which is also adopted in [4]. The 91-image dataset can be decomposed into 24,800 sub-images of size 33×33 for training purpose, which are extracted from original images with a stride of 14. For each LR patch, we subtract its mean and normalize its magnitude, which are later put back to the recovered HR patch. While data augmentation is not adopted in SRCNN, we believe that it plays an important role in training DJSR, to help it focus on meaningful visual features rather than artifacts in training images. We add the following distortions to training images:

- *Translation:* random x-y shifts between [-4, 4] pixels.
- *Rotation:* affine transform with random parameters.
- *Zoom:* random scaling factors between [1/1.2, 1.2]. Note it has to keep the ratio s unchanged.

SDCAE can also be viewed as a data augmentation way by adding noise. We train SDCAE on sub-images, using stochastic gradient descent with a constant momentum of 0.9, and a learning rate η_p of 0.01 (we do not anneal it through training). Mean Squared Error (MSE) is used as the loss function.

Since cross-scale self similarity performs best at small scales [7], we stick to a small upscaling factor s (1.2 by default) for model training, unless otherwise specified. To achieve any targeted upscaling factor s_t, we zoom up an image repeatedly using the learned DJSR model until it is at least as large as the desired size. Then a bicubic interpolation is used to downscale it to the target resolution if necessary. We do not conduct extra joint optimization on the resulting network cascade. The proposed networks are implemented using the CUDA ConvNet package [10] and the ANNA open source library [15], and run on a workstation with 12 Intel Xeon 2.67GHz CPUs and 1 GTX680 GPU.

For color images, we apply SR algorithms to the illuminance channel only, while interpolating the color layers (Cb, Cr) using plain bi-cubic interpolation. However our model is flexible to process more color channels without altering the network design. To avoid border effects, all the convolutional layers have no padding, and the network produces a smaller central output [4].

5.2. Model Analysis

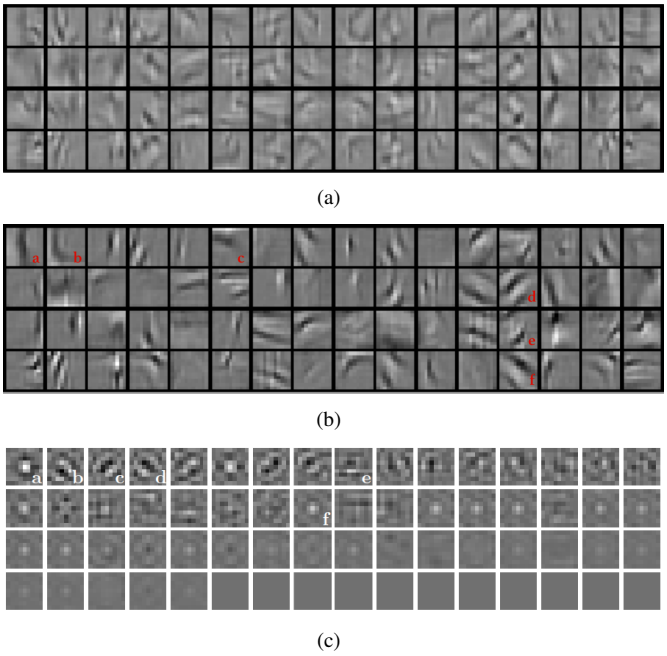

Figure 2. (a) the first layer convolutional filters learnt by SDCAE without any data augmentation; (b) the first layer convolutional filters learnt by SDCAE with data augmentations; (c) the first layer convolutional filters in SRCNN (from the original Fig. 5 in [4]).

Validating Pre-Training We visualize the learned convolutional filters in the first layers of SDCAEs without and with augmentations. They are trained with a relatively large scaling factor $s = 2$, so as to be compared with the first-layer filter visualizations of SRCNN, as depicted in Fig. 2. The training process takes around 7 hours. Both SDCAEs and

SRCNN have 64 channels of 9×9 convolutional filters in the first layer. While there is hardly any recognizable structural features from the filters in (a), the introduction of data augmentations leads to much more clear and interpretable filter responses in (b), from simple edge (curve) detectors at different directions (e.g., a, b and c), to more sophisticated texture descriptors (e.g., d, e and f). On the other hand, since the first layer of SRCNN is designed for patch extraction and representation, it is natural that its learned filters show different from ours. One interesting observation is that SRCNN suffers from several "dead" filters, whose weights are all nearly zeros (as discussed in [4]), whereas almost all filters of SDCAE are fairly strong and diverse. Further, the SDCAE with augmentations obtains an average PSNR of 36.43 dB when testing on the Set 5 [4] (with no fine-tuning applied), where we see a notable performance improvement of 1.44dB compared to the case without augmentations (35.01 dB).

Understanding Fine-Tuning During fine-tuning, we sample LR patches from the input $\mathbf{Y}_{ij}$ with a default size of 15×15 and stride of 1. In this section, we take the *Baby* LR image of size 256×256 for example,which will result in 58,564 patches. Assuming SDCAE has been pre-trained, by default, we fix N = 5 (defined in Algorithm 1), which means the hierarchy will contain 2 upscaled layers (by factors of 1.2 and 1.44) and two downscaled layers (by factors of 1/1.2 and 1/1.44). Fine-tuning a trained SDCAE on *Baby* takes less than 1 hour, and it could be potentially accelerated to a large extent by avoiding working on those homogeneous regions [24]. We will then investigate the influences of the parameter m (that controls the amount of self-example pairs), and the effects of tuning the learning rate η_f, as well as the effects of WBP algorithm.

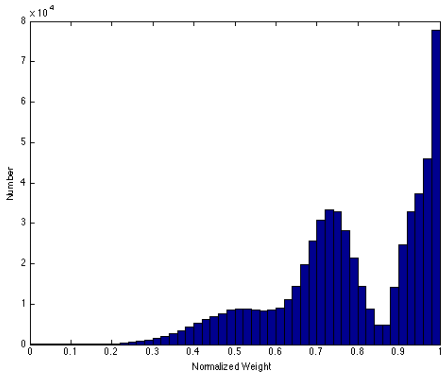

Figure 3. The histogram of normalized weight values of all self-example pairs obtained on *Baby* (m=8).

Fig. 3 depicts the distribution of normalized weight values of all self-example pairs obtained on *Baby*, with m = 8. Notably, two peaks appear on the histogram, one largest peak near the weight value of 1, and the other lower one is centered around 0.75. Further observations reveal that the first peak corresponds to those in-place self examples as in [24], which follow the local scale invariance property and are usually very accurate matches. The second peak, with relatively larger errors, mostly corresponds to the non-local similar examples [7].

Table 1. The effects of m on the average effective volume and the final PSNR (dB) after fine-tuning, when upscaling *Baby* image for 2 times (PSNR is 37.91dB before fine-tuning).

m	4	8	12	16
$\frac{m}{N-1}$	1	2	3	4
V	234,256	468,512	702,768	937,024
V_e	178, 597	365,251	397,242	400,272
V_e^a	0.7624	0.7796	0.5653	0.4272
PNSR	38.01	38.87	38.90	38.91
PNSR$_{NW}$	38.03	38.41	38.00	37.51

Table 2. The effects of η_f on the final PSNR (dB) after fine-tuning, when upscaling *Baby* image for 2 times (The PSNR is 37.91dB before fine-tuning).

η_f	0.01	0.1	0.3	0.5	0.6	0.8
PNSR	37.91	38.12	38.44	38.87	38.36	37.99

To further understand how the amount of self examples and their weights influence the fine-tuning, we introduce several measurements: Let V denote the *total volume* of self-example pairs (thus $V = 58,564 \times m$). Define the *effective volume* V_e as the (rounded) summations of all normalized weights, and the *average effective volume* $V_e^a = \frac{V_e}{V}$. V_e^a can be viewed as an indicator on how reliable and representative the chosen self-example set is. As shown in Table 1, with m growing from 4 to 8, both V_e and V_e^a increase, implying that self similarity is better exploited. The PSNR improvement after fine-tuning also becomes more substantial. However, when m continues going up, both V_e and PSNR reach the plateau, whereas V_e^a decreases dramatically. That clearly manifests that little self similarity information remains to be excavated, and the self example sets assumably turn redundant. Therefore, m is set as 8 by default hereinafter. The last row of Table 1 lists the PSNR results obtained from fine-tuning with standard back propagation, denoted as PNSR$_{NW}$. It is noteworthy that without taking the confidence weights into consideration, more self-example pairs may even harm the SR performance of the pre-trained model when V_e^a drops. The WBP algorithm shows quite robust when more self examples are used in fine-tuning.

Table 2 examines the PNSR changes with varying η_f. We observe that a small learning rate (such as 0.01) does not bring any notable improvement to final results. Until

(a) SRCNN (b) DNC (c) DJSR

Figure 4. 3× SR results of the *Baby* image by: (a) SRCNN, PSNR = 29.22 dB, SSIM = 0.9047; (b) DNC, PSNR = 26.65 dB, SSIM = 0.8490; (c) DJSR, PSNR = 28.74 dB, SSIM = 0.9074.

η_f = 0.5 (the empirical default value), a growing η_f leads to a monotone increase in final PSNR results. That is interpretable, as self examples in any way do not contain as sufficient and diverse information as external examples do; a large learning rate can strengthen their influences on the pre-trained model. It may also help overcome some local minima. However, when η_f is further improved beyond 0.5, the performances start to be undermined, and more fluctuations are observed during the convergence process (η_f = 0.8 actually does not lead to a stable convergence).

5.3. Comparison with State-of-the-Arts

We first compare DJSR qualitatively with two recent DL-based SR methods, SRCNN [4] [2] and DNC [2] [3]. Fig. 4, 5, and 6 demonstrate visual comparisons on three natural images, *Baby*, *Roman*, and *Train*, respectively; all are upsampled by a factor s_t of 3. The zoomed regions are also displayed. SCRNN performs reasonably well on *Baby* and *Train* images, but are visually worse than DNC on the *Roman* image, since *Roman* are abundant in repeating textures on the pillars of the Parthenon, making self similarity especially powerful. DJSR produces image patterns with shaper boundaries and richer textures (see the zoomed pillar regions on *Roman*, and the numbers on *Train*), and suppresses the jaggy and blockiness artifacts discernibly better.

PSNR and SSIM [18] are used to evaluate the performances quantitatively (only the luminance channel is considered). While all three deep networks are optimized under a MSE (equivalent to PSNR) loss, DJSR is slightly worse than SRCNN on *Baby* in terms of PSNR, but obtains the best performances on both *Roman* and *Train* images. What is more, we notice that DJSR is particularly more favorable by SSIM, which measures image quality more consistently with human perception than PSNR. The observation is further verified on the commonly-adopted Set 5 and Set 14 datasets. Such an advantage can be owed to our fine-tuning step, which further enhances the generic model by exploiting the self similar structures of the input. Table 3 compares the average PSNR and SSIM results of the DJSR and SRCNN [4], as well as a few other classical non-DL SR methods, on the Set 5 and Set 14 datasets. DJSR obtains an overall competitive performance, and especially gains a consistent advantage over others in SSIM.

5.4. Evaluation of Sub-models

While the DJSR model could very well compete against the state-of-the-arts, there is still potential room for improvements, by training a group of sub-models and selecting the optimal sub-models for each patch. The number of clusters K is a parameter to be pre-determined. Specifically, we train the sub-models under different K values, and applied them to upscale LR images in Set 5 (s_t = 2). Fig. 7 records the average PSNR values (the black dash line denotes the original DJSR model, i.e., K=1). We can see that when K increases from 50 to 500, the PSNR results gradually raise, as each sub-model is developed to describe a smaller subset of similar image patches more precisely. Yet a slight performance drop occurs at K =800 and continues when K increases to 1,000. A closer look into the clusters demonstrates that when K becomes too large, many clusters contain only as few as thousands of examples, which are inadequate for training a deep network. Such "chaos" sub-models will finally hamper the overall performance.

[2]Results by using the original implementation available at: http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html

[3]Results provided by the authors: http://vipl.ict.ac.cn/paperpage/DNC

[4]DNC is not included, since neither the original codes nor any reported result on the two sets are unavailable. A part of data in Table 3 is from [4]

(a) SRCNN (b) DNC (c) DJSR

Figure 5. 3× SR results of the *Roman* image by: (a) SRCNN, PSNR = 29.97 dB, SSIM = 0.9250; (b) DNC, PSNR = 30.08 dB, SSIM = 0.9293; (c) DJSR, PSNR = 30.69 dB, SSIM = 0.9337.

(a) SRCNN (b) DNC (c) DJSR

Figure 6. 3× SR results of the *Train* image by: (a) SRCNN, PSNR = 29.67 dB, SSIM = 0.9614; (b) DNC, PSNR = 28.02 dB, SSIM = 0.9392; (c) DJSR, PSNR = 30.57 dB, SSIM = 0.9706.

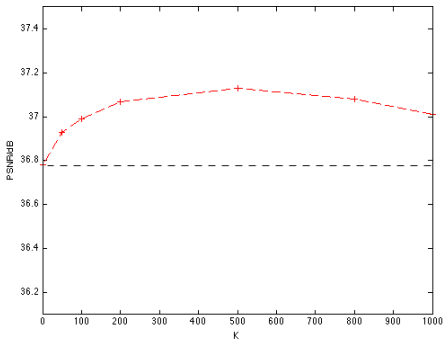

Figure 7. The average PSNR results on Set 5 for upscaling by 2 times, with varying K.

Fig. 7 reminds us that while dividing sub-models is usually supposed to be helpful, an improper choice of K can impact the results negatively. The optimal selection of K is a nontrivial task, and is subject to the bias and variance tradeoff [5]. If K is too small, the boundaries between clusters will be smoothed out and the distinctiveness of sub-models are compromised. On the other hand, an overly larger K will make a part of sub-models unreliable. A simple heuristics is adopted in our experiments: the training dataset is first partitioned into 100 clusters; next, those fragment small clusters (e.g., containing less than 500 samples) are merged into their neighboring clusters. That will usually lead to K between [40, 50]. We find that strategy leads to a good and stable performance improvement, after we try it on several different training sets.

6. Conclusion

In this paper, we investigate a deep joint super resolution (DJSR) model, to exploit external and self similarities for image SR in a unified framework. We utilize external examples to pre-train the model, and fine-tune it using sufficient self examples weighted by their reliability. We thoroughly analyze the model and interpret its behaviors. DJSR is compared with several state-of-the-art SR methods (both DL and non-DL) in our experiments, and shows a visible performance advantage both quantitatively and perceptually. Similar approaches can be extended to many other image restoration applications.

References

[1] H. C. Burger, C. Schuler, and S. Harmeling. Learning how to combine internal and external denoising methods. In *Pattern Recognition*, pages 121–130. Springer, 2013. 1

[2] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen. Deep network cascade for image super-resolution. In *ECCV*, pages 49–64. 2014. 1, 2, 6

Table 3. Average PSNR (dB) and SSIM performances comparisons on the Set 5 and Set 14 datasets

		Bicubic	Sparse Coding [26]	Freedman et.al. [7]	A+ [16]	SRCNN [4]	DJSR
Set 5, s_t*=2*	PSNR	33.66	35.27	33.61	36.24	36.66	**36.78**
	SSIM	0.9299	0.9540	0.9375	0.9544	0.9542	**0.9550**
Set 5, s_t*=3*	PSNR	30.39	31.42	30.77	32.59	**32.75**	32.65
	SSIM	0.8682	0.8821	0.8774	0.9088	0.9090	**0.9161**
Set 14, s_t*=2*	PSNR	30.23	31.34	31.99	**32.58**	32.45	32.51
	SSIM	0.8687	0.8928	0.8921	0.9056	0.9067	**0.9097**
Set 14, s_t*=3*	PSNR	27.54	28.31	28.26	29.13	29.60	**29.96**
	SSIM	0.7736	0.7954	0.8043	0.8188	0.8215	**0.8229**

[3] D. Dai, R. Timofte, and L. Van Gool. Jointly optimized regressors for image super-resolution. 4

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. 2014. 1, 2, 4, 5, 6, 8

[5] W. Dong, D. Zhang, G. Shi, and X. Wu. Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *Image Processing, IEEE Transactions on*, 20(7):1838–1857, 2011. 1, 2, 4, 7

[6] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, 2006. 3

[7] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Transactions on Graphics*, 30(2):12, 2011. 1, 3, 4, 5, 8

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, pages 349–356. IEEE, 2009. 1

[9] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of ICML*, pages 513–520, 2011. 2

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1, 4

[11] A. Liu and B. Ziebart. Robust classification under sample selection bias. In *NIPS*, pages 37–45, 2014. 3

[12] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *ICANN*, pages 52–59. 2011. 2

[13] R. Memisevic, K. Konda, and D. Krueger. Zero-bias autoencoders and the benefits of co-adapting features. *arXiv preprint arXiv:1402.3337*, 2014. 2

[14] I. Mosseri, M. Zontak, and M. Irani. Combining the power of internal and external denoising. In *ICCP*, pages 1–9. IEEE, 2013. 1

[15] T. Paine, P. Khorrami, W. Han, and T. S. Huang. An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*, 2014. 2, 4

[16] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. 8

[17] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010. 2

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004. 6

[19] Z. Wang, Z. Wang, S. Chang, J. Yang, and T. Huang. A joint perspective towards image super-resolution: Unifying external-and self-examples. In *WACV*, pages 596–603. IEEE, 2014. 1

[20] Z. Wang, Y. Yang, Z. Wang, S. Chang, J. Yang, and T. S. Huang. Learning super-resolution jointly from external and internal examples. *arXiv preprint arXiv:1503.01138*, 2015. 1, 2

[21] Z. Wang, Y. Yang, J. Yang, and T. S. Huang. Designing a composite dictionary adaptively from joint examples. *arXiv preprint arXiv:1503.03621*, 2015. 1

[22] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, pages 341–349, 2012. 2

[23] C.-Y. Yang, J.-B. Huang, and M.-H. Yang. Exploiting self-similarities for single frame super-resolution. In *ACCV*, pages 497–510. 2011. 1

[24] J. Yang, Z. Lin, and S. Cohen. Fast image super-resolution based on in-place example regression. In *CVPR*, pages 1059–1066. IEEE, 2013. 1, 2, 3, 5

[25] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang. Coupled dictionary training for image super-resolution. *Image Processing, IEEE Transactions on*, 21(8):3467–3478, 2012. 1, 2

[26] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, 19(11):2861–2873, 2010. 1, 4, 8

[27] M. Zeiler, G. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, pages 2018–2025. IEEE, 2011. 2

[28] M. Zontak and M. Irani. Internal statistics of a single natural image. In *CVPR*, pages 977–984. IEEE, 2011. 1