D^3: Deep Dual-Domain Based Fast Restoration of JPEG-Compressed Images

Zhangyang Wang†, Ding Liu†, Shiyu Chang†, Qing Ling‡, Yingzhen Yang†, and Thomas S. Huang†*
†Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
‡Department of Automation, University of Science and Technology of China, Hefei, 230027, China
{zwang119, dingliu2, chang87, yyang58, t-huang1}@illinois.edu qingling@mail.ustc.edu.cn

Abstract

In this paper, we design a Deep Dual-Domain ($\mathbf{D}^3$) based fast restoration model to remove artifacts of JPEG compressed images. It leverages the large learning capacity of deep networks, as well as the problem-specific expertise that was hardly incorporated in the past design of deep architectures. For the latter, we take into consideration both the prior knowledge of the JPEG compression scheme, and the successful practice of the sparsity-based dual-domain approach. We further design the One-Step Sparse Inference (1-SI) module, as an efficient and light-weighted feed-forward approximation of sparse coding. Extensive experiments verify the superiority of the proposed D^3 model over several state-of-the-art methods. Specifically, our best model is capable of outperforming the latest deep model for around 1 dB in PSNR, and is 30 times faster.

1. Introduction

In visual communication and computing systems, the most common cause of image degradation is arguably compression. Lossy compression, such as JPEG [25] and HEVC-MSP [4], is widely adopted in image and video codecs for saving both bandwidth and in-device storage. It exploits inexact approximations for representing the encoded content compactly. Inevitably, it will introduce undesired complex artifacts, such as blockiness, ringing effects, and blurs. They are usually caused by the discontinuities arising from batch-wise processing, the loss of high-frequency components by coarse quantization, and so on. These artifacts not only degrade perceptual visual quality, but also adversely affect various low-level image processing routines that take compressed images as input [11].

As practical image compression methods are not information theoretically optimal [24], the resulting compression code streams still possess residual redundancies, which makes the restoration of the original signals possible. Different from general image restoration problems, compression artifact restoration has problem-specific properties that can be utilized as powerful priors. For example, JPEG compression first divides an image into 8 × 8 pixel blocks, followed by discrete cosine transformation (DCT) on every block. Quantization is applied on the DCT coefficients of every block, with pre-known quantization levels [25]. Moreover, the compression noises are more difficult to model than other common noise types. In contrast to the tradition of assuming noise to be white and signal independent [2], the non-linearity of quantization operations makes quantization noises non-stationary and signal-dependent.

Various approaches have been proposed to suppress compression artifacts. Early works [6, 22] utilized filtering-based methods to remove simple artifacts. Data-driven methods were then considered to avoid inaccurate empirical modeling of compression degradations. Sparsity-based image restoration approaches have been discussed in [7, 8, 19, 23, 26] to produce sharpened images, but they are often accompanied with artifacts along edges, and unnatural smooth regions. In [24], Liu et.al. proposed a sparse coding process carried out jointly in the DCT and pixel domains, to simultaneously exploit residual redundancies of JPEG code streams and sparsity properties of latent images. More recently, Dong et. al. [11] first introduced deep learning techniques [21] into this problem, by specifically adapting their SR-CNN model in [12]. However, it does not incorporate much problem-specific prior knowledge.

The time constraint is often stringent in image or video codec post-processing scenarios. Low-complexity or even real-time attenuation of compression artifacts is highly desirable [28]. The inference process of traditional approaches, for example, sparse coding, usually involves iterative optimization algorithms, whose inherently sequential structure as well as the data-dependent complexity and latency often constitute a major bottleneck in the computational efficiency [14]. Deep networks benefit from the feed-forward structure and enjoy much faster inference. However, to maintain their competitive performances, deep

*Zhangyang Wang and Thomas Huang's research works are supported in part by US Army Research Office under grant number W911NF-15-1-0317. Qing Ling's research is supported by CAS grant XDA06011203.

networks show demands for increased width (numbers of filters) and depth (number of layers), as well as smaller strides, all leading to growing computational costs [16].

In the paper, we focus on removing artifacts in JPEG compressed images. Our major innovation is to explicitly combine both **the prior knowledge in the JPEG compression scheme** and **the successful practice of dual-domain sparse coding** [24], for designing a task-specific deep architecture. Furthermore, we introduce a One-Step Sparse Inference **(1-SI)** module, that acts as a highly efficient and light-weighted approximation of the sparse coding inference [10]. 1-SI also reveals important inner connections between sparse coding and deep learning. The proposed model, named Deep Dual-Domain (**D^3**) based fast restoration, proves to be more effective and interpretable than general deep models. It gains remarkable margins over several state-of-the-art methods, in terms of both **restoration performance** and **time efficiency**.

2. Related Work

Our work is inspired by the prior wisdom in [24]. Most previous works restored compressed images in either the pixel domain [2] or the DCT domain [25] solely. However, an isolated quantization error of one single DCT coefficient is propagated to all pixels of the same block. An aggressively quantized DCT coefficient can further produce structured errors in the pixel-domain that correlate to the latent signal. On the other hand, the compression process sets most high frequency coefficients to zero, making it impossible to recover details from only the DCT domain. In view of their complementary characteristics, the dual-domain model was proposed in [24]. While the spatial redundancies in the pixel domain were exploited by a learned dictionary [2], the residual redundancies in the DCT domain were also utilized to directly restore DCT coefficients. In this way, quantization noises were suppressed without propagating errors. The final objective (see Section 3.1) is a combination of DCT- and pixel-domain sparse representations, which could cross validate each other.

To date, deep learning [21] has shown impressive results on both high-level and low-level vision problems [35, 36]. The SR-CNN proposed by Dong et al. [12] showed the great potential of end-to-end trained networks in image super resolution (SR). Their recent work [11] proposed a four-layer convolutional network that was tuned based on SR-CNN, named Artifacts Reduction Convolutional Neural Networks (AR-CNN), which was effective in dealing with various compression artifacts.

In [14], the authors leveraged fast trainable regressors and constructed feed-forward network approximations of the learned sparse models. By turning sparse coding into deep networks, one may expect faster inference, larger learning capacity, and better scalability. Similar views were adopted in [29] to develop a fixed-complexity algorithm for solving structured sparse and robust low rank models. The paper [17] summarized the methodology of "deep unfolding". [35] proposed deeply improved sparse coding for SR, which can be incarnated as an end-to-end neural network. Lately, [34] proposed Deep ℓ_0 Encoders, to model ℓ_0 sparse approximation as feed-forward neural networks. [33] further extended the same "task-specific" strategy to graph-regularized ℓ_1 approximation. Our task-specific architecture shares similar spirits with these works.

3. Deep Dual-Domain (D^3) based Restoration

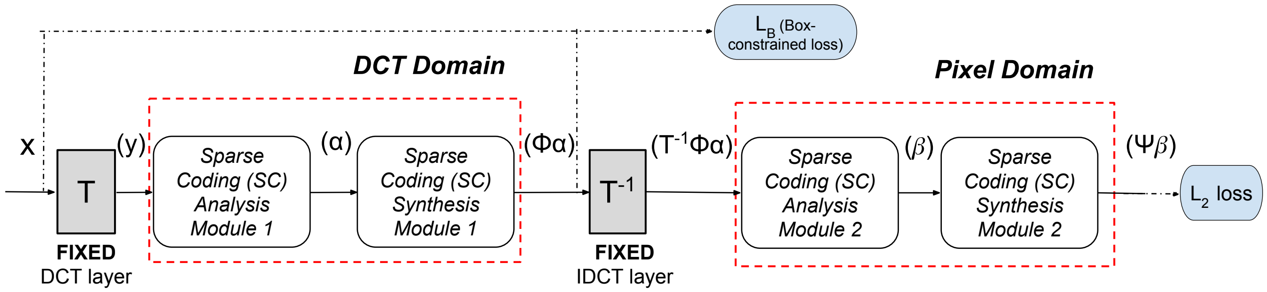
3.1. Sparsity-based Dual-Domain Formulation

We first review the sparsity-based dual-domain restoration model established in [24]. Considering a training set of **uncompressed** images, pixel-domain blocks $\{\hat{x}_i\} \in R^m$ (vectorized from a $\sqrt{m} \times \sqrt{m}$ patch; m = 64 for JPEG) are drawn for training, along with their **quantized** DCT coefficient blocks $\{y_i\} \in R^m$. For each (JPEG-coded) input $x_t \in R^m$, two dictionaries $\mathbf{\Phi} \in R^{m \times p_\Phi}$ and $\mathbf{\Psi} \in R^{m \times p_\Psi}$ (p_Φ and p_Ψ denote the dictionary sizes) are constructed from training data $\{y_i\}$ and $\{\hat{x}_i\}$, in the DCT and pixel domains, respectively, via locally adaptive feature selection and projection. The following optimization model is then solved during the testing stage:

$$\begin{aligned} \min_{\{\alpha,\beta\}} &\ ||y_t - \mathbf{\Phi}\alpha||_2^2 + \lambda_1||\alpha||_1 \\ &+\lambda_2||T^{-1}\mathbf{\Phi}\alpha - \mathbf{\Psi}\beta||_2^2 + \lambda_3||\beta||_1, \\ &s.t. \quad q^L \preceq \mathbf{\Phi}\alpha \preceq q^U. \end{aligned} \tag{1}$$

where $y_t \in R^m$ is the DCT coefficient block for x_t. $\alpha \in R^{p_\Phi}$ and $\beta \in R^{p_\Psi}$ are sparse codes in the DCT and pixel domains, respectively. T^{-1} denotes the inverse discrete cosine transform (IDCT) operator. λ_1, λ_2 and λ_3 are positive scalars. One noteworthy point is the inequality constraint, where q^L and q^U represents the (pre-known) quantization intervals according to the JPEG quantization table [25]. The constraint incorporates the important side information and further confines the solution space. Finally, $\mathbf{\Psi}\beta$ provides an estimate of the original uncompressed pixel block $\hat{x}_t$.

Such a sparsity-based dual-domain model (1) exploits residual redundancies (e,g, inter-DCT-block correlations) in the DCT domain without spreading errors into the pixel domain, and at the same time recovers high-frequency information driven by a large training set. However, note that the inference process of (1) relies on iterative algorithms, and is computational expensive. Also in (1), the three parameters λ_1, λ_2 and λ_3 have to be manually tuned. The authors of [24] simply set them all equal, which may hamper the performance. In addition, the dictionaries $\mathbf{\Phi}$ and $\mathbf{\Psi}$ have to be individually learned for each patch, which allows for extra flexibility but also brings in heavy computation load.



L_B (Box-constrained loss)
DCT Domain
Pixel Domain
x
T
(y)
Sparse Coding (SC) Analysis Module 1
(α)
Sparse Coding (SC) Synthesis Module 1
(Φα)
T^-1
(T^-1Φα)
Sparse Coding (SC) Analysis Module 2
(β)
Sparse Coding (SC) Synthesis Module 2
(Ψβ)
L_2 loss
FIXED
DCT layer
FIXED
IDCT layer

Figure 1. The illustration of Deep Dual-Domain ($\mathbf{D^3}$) based model (all subscripts are omitted for simplicity). The black solid lines denote the network inter-layer connections, while the black dash lines connect to the loss functions. The two red dash-line boxes depict the two stages that incorporate DCT and pixel domain sparsity priors, respectively. The two grey blocks denote constant DCT and IDCT layers, respectively. The notations within parentheses along the pipeline are to remind the corresponding variables in (1).

3.2. $\mathbf{D^3}$: A Feed-Forward Network Formulation

In training, we have the compressed pixel-domain blocks $\{x_i\}$, accompanied with the original uncompressed blocks $\{\hat{x}_i\}$. During testing, for an compressed input x_t, our goal is to estimate the original $\hat{x}_t$, using the redundancies in both DCT and pixel domains, as well as JPEG prior knowledge.

As illustrated in Fig. 1, the input x_t is first transformed into its DCT coefficient block y_t, by feeding through the constant 2-D DCT matrix layer T. The subsequent two layers aim to enforce DCT domain sparsity, where we refer to the concepts of analysis and synthesis dictionaries in sparse coding [15]. The Sparse Coding (SC) Analysis Module 1 is implemented to solve the following type of sparse inference problem in the DCT domain (λ is a positive coefficient):

$$\min_\alpha \tfrac{1}{2}||y_t - \mathbf{\Phi}\alpha||_2^2 + \lambda||\alpha||_1. \tag{2}$$

The Sparse Coding (SC) Synthesis Module 1 outputs the DCT-domain sparsity-based reconstruction in (1), i.e., $\mathbf{\Phi}\alpha$.

The intermediate output $\mathbf{\Phi}\alpha$ is further constrained by an auxiliary loss, which encodes the inequality constraint in (1): $q^L \preceq \mathbf{\Phi}\alpha \preceq q^U$. We design the following **signal-dependent, box-constrained [20] loss**:

$$L_B(\mathbf{\Phi}\alpha, x) = ||[\mathbf{\Phi}\alpha - q^U(x)]_+||_2^2 + ||[q^L(x) - \mathbf{\Phi}\alpha]_+||_2^2. \tag{3}$$

Note it takes not only $\mathbf{\Phi}\alpha$, but also x as inputs, since the actual JPEG quantization interval $[q^L, q^U]$ depends on x. The operator $[\ \ ]_+$ keeps the nonnegative elements unchanged while setting others to zero. Eqn. (3) will thus only penalize the coefficients falling out of the quantization interval.

After the constant IDCT matrix layer T^{-1}, the DCT-domain reconstruction $\mathbf{\Phi}\alpha$ is transformed back to the pixel domain for one more sparse representation. The SC Analysis Module 2 solves (γ is a positive coefficient):

$$\min_\beta \tfrac{1}{2}||T^{-1}\mathbf{\Phi}\alpha - \mathbf{\Psi}\beta||_2^2 + \gamma||\beta||_1, \tag{4}$$

while the SC Synthesis Module 2 produces the final pixel-domain reconstruction $\mathbf{\Psi}\beta$. Finally, the L_2 loss between $\mathbf{\Psi}\beta$ and $\hat{x}_i$ is enforced.

Note that in the above, we try to correspond the intermediate outputs of $\mathbf{D^3}$ with the variables in (1), in order to help understand the close analytical relationship between the proposed deep architecture with the sparse coding-based model. That does not necessarily imply any exact numerical equivalence, since $\mathbf{D^3}$ allows for end-to-end learning of all parameters (including λ in (2) and γ in (4)). However, we will see in experiments that such enforcement of the specific problem structure improves the network performance and efficiency remarkably. In addition, the above relationships remind us that the deep model could be well initialized from the sparse coding components.

3.3. One-Step Sparse Inference Module

The implementation of SC Analysis and Synthesis Modules appears to be the core of $\mathbf{D^3}$. While the synthesis process is naturally feed-forward by multiplying the dictionary, it is less straightforward to transform the sparse analysis (or inference) process into a feed-forward network.

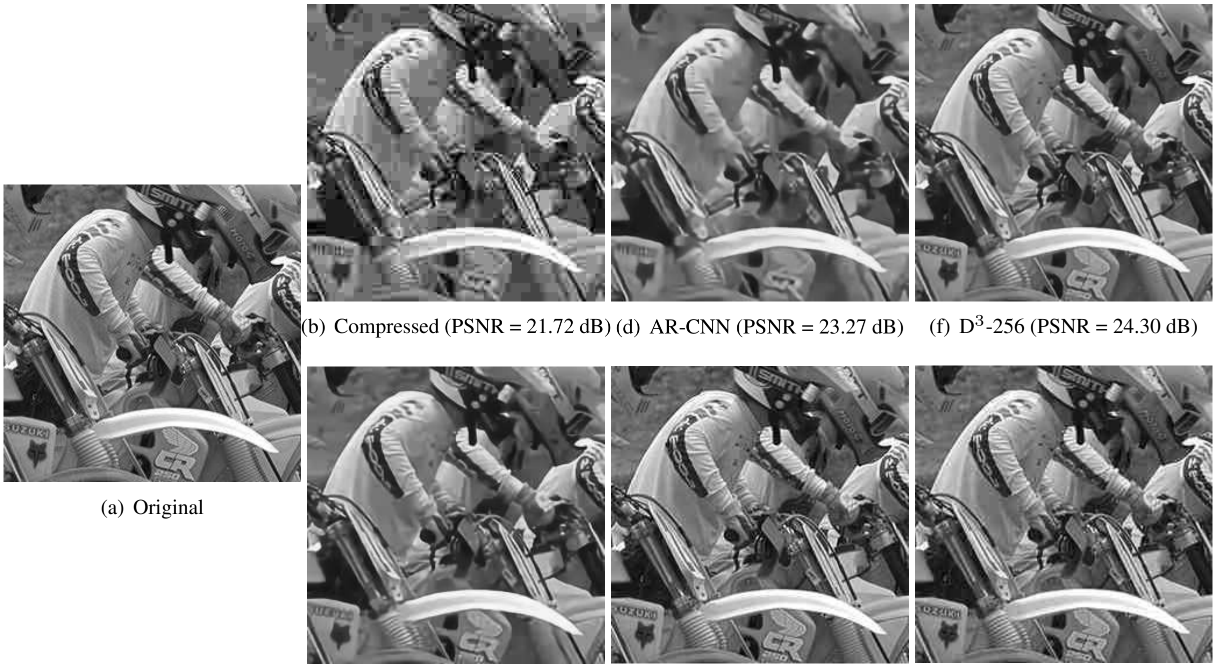
We take (2) as an example, while the same solution applies to (4). Such a sparse inference problem could be solved by the iterative shrinkage and thresholding algorithm (ISTA) [5], each iteration of which updates as follows:

$$\boldsymbol{\alpha}^{k+1} = s_\lambda(\boldsymbol{\alpha}^k + \mathbf{\Phi}^T(y_t - \mathbf{\Phi}\boldsymbol{\alpha}^k)), \tag{5}$$

where $\boldsymbol{\alpha}^k$ denotes the intermediate result of the k-th iteration, and where s_λ is an element-wise shrinkage function ($\mathbf{u}$ is a vector and $\mathbf{u}_i$ is its i-th element, $i = 1, 2, ..., p$):

$$[s_\lambda(\mathbf{u})]_i = \text{sign}(\mathbf{u}_i)[|\mathbf{u}_i| - \lambda_i]_+. \tag{6}$$

The learned ISTA (LISTA) [14] parameterized encoder further proposed a natural network implementation of ISTA. The authors time-unfolded and truncated (5) into a fixed number of stages (more than 2), and then jointly tuned all parameters with training data, for a good feed-forward approximation of sparse inference. The similar unfolding methodology has been lately exploited in [17], [29], [30].



(a) Original

(b) Compressed (PSNR = 21.72 dB)

(c) S-D^2 (PSNR = 22.87 dB)

(d) AR-CNN (PSNR = 23.27 dB)

(e) D^3-128 (PSNR = 23.94 dB)

(f) D^3-256 (PSNR = 24.30 dB)

(g) D-Base-256 (PSNR = 23.48 dB)

Figure 2. Visual comparison of various methods on *Bike* at Q = 5. The corresponding PSNR values (in dB) are also shown.

In our work, we launch a more aggressive approximation, by only keeping one iteration of (5), leading to a One-Step Sparse Inference (**1-SI**) Module. Our major motivation lies in the same observation as in [11] that overly deep networks could adversely affect the performance in low-level vision tasks. Note that we have two SC Analysis modules where the original LISTA applies, and two more SC Synthesis modules (each with one learnable layer). Even only two iterations are kept as in [14], we end up with a six-layer network, that suffers from both difficulties in training [11] and fragility in generalization [31] for this task.

A 1-SI module takes the following simplest form:

$$\boldsymbol{\alpha} = s_\lambda(\mathbf{\Phi} y_t), \tag{7}$$

which could be viewed as first passing through a fully-connected layer ($\mathbf{\Phi}$), followed by neurons that take the form of s_λ. We further rewrite (6) as [35] did[1]:

$$[s_\lambda(\mathbf{u})]_i = \lambda_i \cdot \text{sign}(\mathbf{u}_i)(|\mathbf{u}_i|/\lambda_i - 1)_+ = \lambda_i s_1(\mathbf{u}_i/\lambda_i) \tag{8}$$

Eqn. (8) indicates that the original neuron with trainable thresholds can be decomposed into two linear scaling layers plus a unit-threshold neuron. The weights of the two scaling layers are diagonal matrices defined by $\boldsymbol{\theta}$ and its element-wise reciprocal, respectively. The unit-threshold neuron s_1 could in essence be viewed as a double-sided and translated variant of ReLU [21].

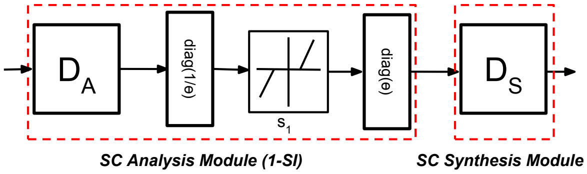


Figure 3. The illustration of SC Analysis and Synthesis Modules. The former is implemented by the proposed 1-SI module (7). Both D_A and D_S are fully-connected layers, while diag(θ) and diag($1/\theta$) denotes the two diagonal scaling layers.

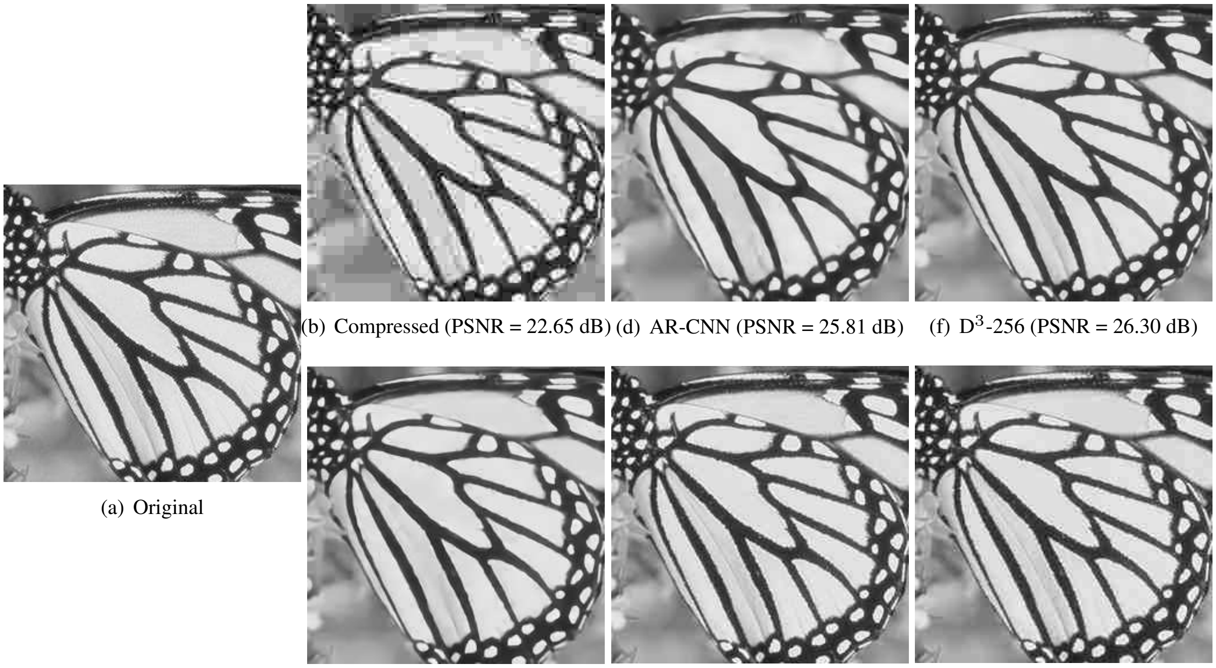
A related form to (7) was obtained in [10] on a different case of non-negative sparse coding. The authors studied its connections with the soft-threshold feature for classification, but did not correlate it with network architectures.

3.4. Model Overview

By plugging in the 1-SI module (7), we are ready to obtain the SC Analysis and Synthesis Modules, as in Fig. 3. By comparing Fig. 3 with Eqn. (2) (or (4)), it is easy to notice the analytical relationships between D_A and $\mathbf{\Phi}^T$ (or $\mathbf{\Psi}^T$), D_S and $\mathbf{\Phi}$ (or $\mathbf{\Psi}$), as well as θ and λ (or γ). In fact, those network hyperparamters could be well initialized from the sparse coding parameters, which could be obtained easily. The entire D^3 model, consisting of four learnable fully-connected weight layers (except for the diagonal layers), are then trained from end to end [2].

[1] In (8), we slightly abuse notations, and set λ to be a vector of the same dimension as $\mathbf{u}$, in order for extra element-wise flexibility.

[2] From the analytical perspective, $\mathbf{D}_S$ is the transpose of $\mathbf{D}_A$, but we untie them during training for larger learning capability.



(a) Original

(b) Compressed (PSNR = 22.65 dB) (d) AR-CNN (PSNR = 25.81 dB) (f) D^3-256 (PSNR = 26.30 dB)

(c) S-D^2 (PSNR = 24.87 dB) (e) D^3-128 (PSNR = 24.74 dB) (g) D-Base-256 (PSNR = 24.28 dB)

Figure 4. Visual comparison of various methods on *Monarch* at Q = 5. The corresponding PSNR values (in dB) are also shown.

In Fig. 3, we intentionally do not combine θ into $\mathbf{D}_A$ layer (also $1/\theta$ into $\mathbf{D}_S$ layer), for the reason that we still wish to keep θ and $1/\theta$ layers tied as element-wise reciprocal. That proves to have positive implications in our experiments. If we absorb the two diagonal layers into $\mathbf{D}_A$ and $\mathbf{D}_S$, Fig. 3 is reduced to two fully connected weight matrices, concatenated by one layer of hidden neurons (8). However, keeping the "decomposed" model architecture facilitates the incorporation of problem-specific structures.

3.5. Complexity Analysis

From the clear correspondences between the sparsity-based formulation and the D^3 model, we immediately derive the dimensions of weight layers, as in Table 1.

Table 1. Dimensions of all layers in the D^3 model

Layer	$\mathbf{D}_A$	$\mathbf{D}_S$	diag(θ)
Stage I (DCT Domain)	$p_\Phi \times m$	$m \times p_\Phi$	p_Φ
Stage II (Pixel Domain)	$p_\Psi \times m$	$m \times p_\Psi$	p_Ψ

3.5.1 Time Complexity

During training, deep learning with the aid of gradient descent scales linearly in time and space with the number of training samples. We are primarily concerned with the time complexity during testing (inference), which is more relevant to practical usages. Since all learnable layers in the D^3 model are fully-connected, the inference process of D^3 is nothing more than a series of matrix multiplications. The multiplication times are counted as: $p_\Phi m$ (D_A in Stage I) + $2p_\Phi$ (two diagonal layers) + $p_\Phi m$ (D_S in Stage I) + $p_\Psi m$ (D_A in Stage II) + $2p_\Psi$ (two diagonal layers) + $p_\Psi m$ (D_S in Stage II). The 2D DCT and IDCT each takes $\frac{1}{2}m\log(m)$ multiplications [25] . Therefore, the total inference time complexity of D^3 is:

$$C_{D^3} = 2(p_\Phi + p_\Psi)(m+1) + m\log(m) \approx 2m(p_\Phi + p_\Psi). \tag{9}$$

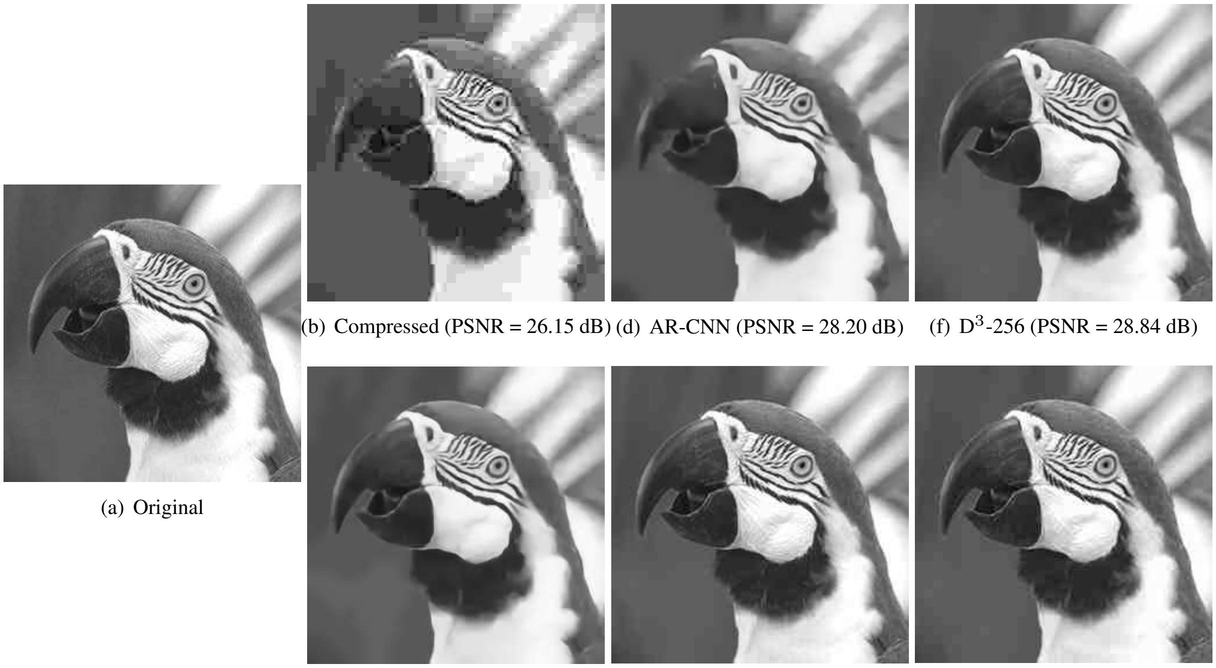
The complexity could also be expressed as $O(p_\Phi + p_\Psi)$.

It is obvious that the sparse coding inference [24] has dramatically higher time complexity. We are also interested in the inference time complexity of other competitive deep models, especially AR-CNN [11]. For their fully convolutional architecture, the total complexity [16] is:

$$C_{conv} = \textstyle\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2, \tag{10}$$

where l is the layer index, d is the total depth, n_l is the number of filters in the l-th layer, s_l is the spatial size of the filter, and m_l is the spatial size of the output feature map.

The theoretical time complexities in (9) and (10) do not represent the actual running time, as they depend on different configurations and can be sensitive to implementations and hardware. Yet, our actual running time scales nicely with those theoretical results.



(b) Compressed (PSNR = 26.15 dB) (d) AR-CNN (PSNR = 28.20 dB) (f) D^3-256 (PSNR = 28.84 dB)

(a) Original

(c) S-D^2 (PSNR = 27.92 dB) (e) D^3-128 (PSNR = 27.52 dB) (g) D-Base-256 (PSNR = 27.21 dB)

Figure 5. Visual comparison of various methods on *Parrots* at Q = 5. The corresponding PSNR values are also shown.

3.5.2 Parameter Complexity

The total number of free parameters in D^3 is:

$$N_{D^3} = 2p_\Phi m + p_\Phi + 2p_\Psi m + p_\Psi = 2(p_\Phi + p_\Psi)(m+1). \tag{11}$$

As a comparison, the AR-CNN model [11] contains:

$$N_{conv} = \textstyle\sum_{l=1}^{d} n_{l-1} \cdot n_l \cdot s_l^2. \tag{12}$$

4. Experiments

4.1. Implementation and Setting

We use the disjoint training set (200 images) and test set (200 images) of BSDS500 database [3], as our training set; its validation set (100 images) is used for validation, which follows [11]. For training the D^3 model, we first divide each original image into overlapped 8×8 patches, and subtract the pixel values by 128 as in the JPEG mean shifting process. We then perform JPEG encoding on them by MATLAB JPEG encoder with a specific quality factor Q, to generate the corresponding compressed samples. Whereas JPEG works on non-overlapping patches, we emphasize that the training patches are overlapped and extracted from arbitrary positions. For a testing image, we sample 8×8 blocks with a stride of 4, and apply the D^3 model in a patch-wise manner. For a patch that misaligns with the original JPEG block boundaries, we find its most similar coding block from its 16×16 local neighborhood, whose quantization intervals are then applied to the misaligned patch. We find this practice effective and important for removing blocking artifacts and ensuring the neighborhood consistency. The final result is obtained via aggregating all patches, with the overlapping regions averaged.

The proposed networks are implemented using the cuda-convnet package [21]. We apply a constant learning rate of 0.01, a batch size of 128, with no momentum. Experiments run on a workstation with 12 Intel Xeon 2.67GHz CPUs and 1 GTX680 GPU. The two losses, L_B and L_2, are equally weighted. For the parameters in Table 1, m is fixed as 64. We try different values of p_Φ and p_Ψ in experiments.

Based on the solved Eqn. (1), one could initialize D_A, D_S, and θ from Φ, Φ^T and λ in the DCT domain block of Fig. 1, and from Ψ, Ψ^T and γ in the pixel domain block, respectively. In practice, we find that such an initialization strategy benefits the performances, and usually leads to faster convergence.

We test the quality factor Q = 5, 10, and 20. For each Q, we train a dedicated model. We further find the easy-hard transfer suggested by [11] useful. As images of low Q values (heavily compressed) contain more complex artifacts, it is helpful to use the features learned from images of high Q values (lightly compressed) as a starting point. In practice, we first train the D^3 model on JPEG compressed images with $Q = 20$ (the highest quality). We then initialize the $Q = 10$ model with the $Q = 20$ model, and similarly, initialize $Q = 5$ model from the $Q = 10$ one.

Table 2. The average results of PSNR (dB), SSIM, PSNR-B (dB) on the LIVE1 dataset.

		Compressed	S-D^2	AR-CNN	D^3-128	D^3-256	D-Base-256
Q = 5	PSNR	24.61	25.83	26.64	26.26	**27.37**	25.83
	SSIM	0.7020	0.7170	0.7274	0.7203	**0.7303**	0.7186
	PSNR-B	22.01	25.64	26.46	25.86	**26.95**	25.51
Q = 10	PSNR	27.77	28.88	29.03	28.62	**29.96**	28.24
	SSIM	0.7905	0.8195	0.8218	0.8198	**0.8233**	0.8161
	PSNR-B	25.33	27.96	28.76	28.33	**29.45**	27.57
Q = 20	PSNR	30.07	31.62	31.30	31.20	**32.21**	31.27
	SSIM	0.8683	0.8830	0.8871	0.8829	**0.8903**	0.8868
	PSNR-B	27.57	29.73	30.80	30.56	**31.35**	29.25
#Param		\	NA	106,448	33, 280	66, 560	66, 560

4.2. Restoration Performance Comparison

We include the following two relevant, state-of-the-art methods for comparison:

- **Sparsity-based Dual-Domain Method (S-D^2)** [24] could be viewed as the "shallow" counterpart of D^3. It has outperformed most traditional methods [24], such as BM3D [9] and DicTV [7], with which we thus do not compare again. The algorithm has a few parameters to be manually tuned. Especially, their dictionary atoms are adaptively selected by a nearest-neighbour type algorithm; the number of selected atoms varies for every testing patch. Therefore, the parameter complexity of S-D^2 cannot be exactly computed.

- **AR-CNN** has been the latest deep model resolving the JPEG compression artifact removal problem. In [11], the authors show its advantage over SA-DCT [13], RTF [18], and SR-CNN [12]. We adopt the default network configuration in [11]: s_1 = 9, s_2 = 7, s_3 = 1, s_4 = 5; n_1 = 64, n_2 = 32, n_3 = 16, n_4 = 1. The authors adopted the easy-hard transfer in training.

For D^3, we test p_Φ = p_Ψ = 128 and 256 [3]. The resulting D^3 models are denoted as D^3-128 and D^3-256, respectively. In addition, to verify the superiority of our task-specific design, we construct a fully-connected Deep Baseline Model (D-Base), of the same complexity with D^3-256, named D-Base-256. It consists of four weight matrices of the same dimensions as D^3-256's four trainable layers[4]. D-Base-256 utilizes ReLU [21] neurons and the dropout technique.

We use the 29 images in the LIVE1 dataset [27] (converted to the gray scale) to evaluate both the quantitative and qualitative performances. Three quality assessment criteria: PSNR, structural similarity (SSIM) [32], and PSNR-B [37], are evaluated, the last of which is designed specifically to assess blocky images. The averaged results on the LIVE1 dataset are list in Table 2.

Compared to S-D^2, both D^3-128 and D^3-256 gain remarkable advantages, thanks to the end-to-end training as deep architectures. As p_Φ and p_Ψ grow from 128 to 256, one observes clear improvements in PSNR/SSIM/PSNR-B. D^3-256 has outperformed the state-of-the-art ARCNN, for around 1 dB in PSNR. Moreover, D^3-256 also demonstrates a notable performance margin over D-Base-256, although they possess the same number of parameters. D^3 is thus verified to benefit from its task-specific architecture inspired by the sparse coding process (1), rather than just the large learning capacity of generic deep models. The parameter numbers of different models are compared in the last row of Table 2. It is impressive to see that D^3-256 also takes less parameters than AR-CNN.

We display three groups of visual results, on *Bike*, *Monarch* and *Parrots* images, when Q = 5, in Figs. 2, 4 and 5, respectively. AR-CNN tends to generate oversmoothness, such as in the edge regions of butterfly wings and parrot head. S-D^2 is capable of restoring sharper edges and textures. The D^3 models further reduce the unnatural artifacts occurring in S-D^2 results. Especially, while D^3-128 results still suffer from a small amount of visible ringing artifacts, D^3-256 not only shows superior in preserving details, but also suppresses artifacts well.

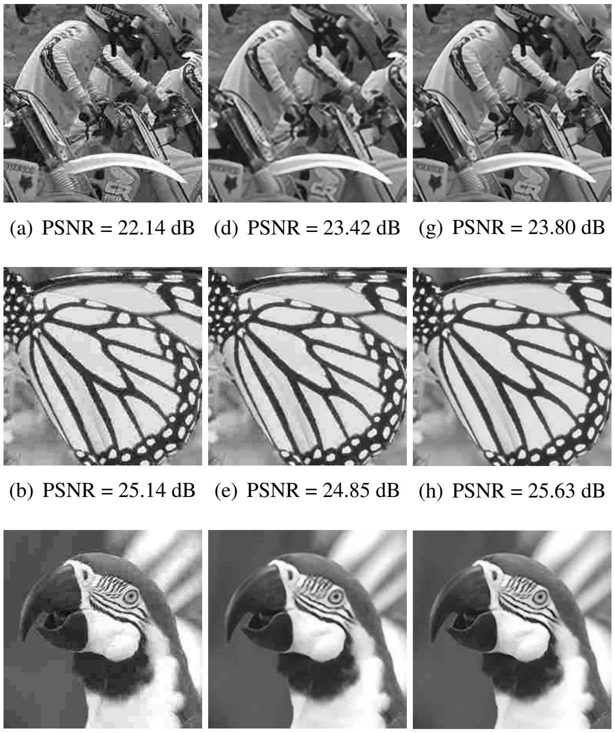
4.3. Analyzing the Impressive Results of D^3

We attribute our impressive recovery of clear fine details, to the combination of our specific pipeline, the initialization, and the box-constrained loss.

Task-specific and interpretable pipeline The benefits of our specifically designed architecture were demonstrated by the comparison experiments to baseline encoders. Further, we provide intermediate outputs of the IDCT layer, i.e., the recovery after the DCT-domain reconstruction. We hope that it helps understand how each component, i.e., the DCT-domain reconstruction or the pixel-domain reconstruction, contributes to the final results. As shown in Fig. 6 (a)-(c),

[3] from the common experiences of choosing dictionary sizes [2]

[4] D-Base-256 is a four-layer neural network, performed on the pixel domain, without DCT/IDCT layers. The diagonal layers contain a very small portion of parameters and are ignored here.



(a) PSNR = 22.14 dB (d) PSNR = 23.42 dB (g) PSNR = 23.80 dB

(b) PSNR = 25.14 dB (e) PSNR = 24.85 dB (h) PSNR = 25.63 dB

(c) PSNR = 26.74 dB (f) PSNR = 27.63 dB (i) PSNR = 28.28 dB

Figure 6. Intermediate and comparison results, on *Bike*, *Monarch*, and *Parrot*, at Q = 5: (a) - (c) the intermediate recovery results after the DCT-domain reconstruction; (d) - (f) the results trained with random initialization; (g) - (i) the results trained without the box-constrained loss. PSNR values are reported.

such intermediate reconstruction results contain both sharpened details (see the characters in (a), which become more recognizable), and unexpected noisy patterns (see (a) (b) (c) for the blockiness, and ringing-type noise along edges and textures). It implies that Stage I DCT-domain reconstruction has enhanced the high-frequency features, yet introducing artifacts simultaneously due to quantization noises. Afterwards, Stage II pixel-domain reconstruction performs extra noise suppression and global reconstruction, which leads to the artifact-free and more visually pleasing final results.

Sparse coding-based initialization We conjecture that the reason why D^3 is more capable in restoring the text on *Bike* and other subtle textures hinges on our sparse coding-based initialization, as an important training detail in D^3. To verify that, we re-train D^3 with random initialization, with the testing results in Fig. 6 (d)-(f), which turn out to be visually smoother (closer to AR-CNN results). For example, the characters in (d) are now hardly recognizable. We notice that the S-D^2 results, as in original Fig. 2-5 (c), also presented sharper and more recognizable texts and details than AR-CNN. These observations validate our conjecture. So the next question is, **why sparse coding helps significantly here**? The quantization process can be considered as as a low-pass filter that cuts off high-frequency information. The dictionary atoms are learned from offline high-quality training images, which contain rich high-frequency information. The sparse linear combination of atoms is thus richer in high-frequency details, which might not necessarily be the case in generic regression (as in deep learning).

Box-constrained loss The loss L_B (3) acts as another effective regularization. We re-train D^3 without the loss, and obtain the results in Fig. 6 (g)-(i). It is observed that the box-constrained loss helps generate details (e.g., comparing characters in (g) with those in Fig. 2 (f)), by bounding the DCT coefficients, and brings PSNR gains.

4.4. Running Time Comparison

The image or video codecs desire highly efficient compression artifact removal algorithms as the post-processing tool. Traditional TV and digital cinema business uses frame rate standards such as 24p (i.e., 24 frames per second), 25p, and 30p. Emerging standards require much higher rates. For example, high-end High-Definition (HD) TV systems adopt 50p or 60p; the Ultra-HD (UHD) TV standard advocates 100p/119.88p/120p; the HEVC format could reach the maximum frame rate of 300p [1]. To this end, higher time efficiency is as desirable as improved performances.

Table 3. Averaged running time comparison (ms) on LIVE1.

	AR-CNN	D^3-128	D^3-256	D-Base-256
Q = 5	396.76	7.62	12.20	9.85
Q = 10	400.34	8.84	12.79	10.27
Q = 20	394.61	8.42	12.02	9.97

We compare the averaged testing times of AR-CNN and the proposed D^3 models in Table 3, on the LIVE29 dataset, using the same machine and software environment. All running time was collected from GPU tests. Our best model, D^3-256, takes approximately 12 ms per image; that is more than **30 times faster** than AR-CNN. The speed difference is NOT mainly caused by the different implementations. Both being completely feed-forward, AR-CNN relies on the time-consuming convolution operations while ours takes only a few matrix multiplications. That is in accordance with the theoretical time complexities computed from (9) and (10), too. As a result, D^3-256 is able to process 80p image sequences (or even higher). To our best knowledge, D^3 is the **fastest** among all state-of-the-art algorithms, and proves to be a practical choice for HDTV industrial usage.

5. Conclusion

We introduce the D^3 model, for the fast restoration of JPEG compressed images. The successful combination of both JPEG prior knowledge and sparse coding expertise has made D^3 highly effective and efficient. In the future, we aim to extend the methodology to more related applications.

References

[1] https://en.wikipedia.org/wiki/Frame_rate/. 8

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *TSP*, 54(11):4311–4322, 2006. 1, 2, 7

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011. 6

[4] E. A. Ayele and S. Dhok. Review of proposed high efficiency video coding (hevc) standard. *International Journal of Computer Applications*, 59(15):1–9, 2012. 1

[5] T. Blumensath and M. E. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6):629–654, 2008. 3

[6] K. Bredies and M. Holler. A total variation-based jpeg decompression model. *SIAM Journal on Imaging Sciences*, 5(1):366–393, 2012. 1

[7] H. Chang, M. K. Ng, and T. Zeng. Reducing artifacts in jpeg decompression via a learned dictionary. *TSP*, 2014. 1, 7

[8] I. Choi, S. Kim, M. S. Brown, and Y.-W. Tai. A learning-based approach to reduce jpeg artifacts in image matting. In *ICCV*, pages 2880–2887. IEEE, 2013. 1

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *TIP*, 16(8):2080–2095, 2007. 7

[10] M. Denil and N. de Freitas. Recklessly approximate sparse coding. *arXiv preprint arXiv:1208.0959*, 2012. 2, 4

[11] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. *ICCV*, 2015. 1, 2, 4, 5, 6, 7

[12] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014. 1, 2, 7

[13] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *TIP*, 2007. 7

[14] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *ICML*, pages 399–406, 2010. 1, 2, 3, 4

[15] S. Gu, L. Zhang, W. Zuo, and X. Feng. Projective dictionary pair learning for pattern classification. In *NIPS*, pages 793–801, 2014. 3

[16] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *CVPR*, 2015. 2, 5

[17] J. R. Hershey, J. L. Roux, and F. Weninger. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv preprint arXiv:1409.2574*, 2014. 2, 4

[18] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *ECCV*, pages 112–125. Springer, 2012. 7

[19] C. Jung, L. Jiao, H. Qi, and T. Sun. Image deblocking via sparse representation. *Signal Processing: Image Communication*, 27(6):663–677, 2012. 1

[20] D. Kim, S. Sra, and I. S. Dhillon. Tackling box-constrained optimization via a new projected quasi-newton approach. *SIAM Journal on Scientific Computing*, 2010. 3

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1, 2, 4, 6, 7

[22] K. Lee, D. S. Kim, and T. Kim. Regression-based prediction for blocking artifact reduction in jpeg-compressed images. *TIP*, 14(1):36–48, 2005. 1

[23] X. Liu, G. Cheung, X. Wu, and D. Zhao. Inter-block soft decoding of jpeg images with sparsity and graph-signal smoothness priors. In *ICIP*. IEEE, 2015. 1

[24] X. Liu, X. Wu, J. Zhou, and D. Zhao. Data-driven sparsity-based restoration of jpeg-compressed images in dual transform-pixel domain. In *CVPR*, 2015. 1, 2, 5, 7

[25] W. B. Pennebaker and J. L. Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1993. 1, 2, 5

[26] R. Rothe, R. Timofte, and L. Van Gool. Efficient regression priors for reducing image compression artifacts. In *IEEE ICIP*, 2015. 1

[27] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik. Live image quality assessment database release 2, 2005. 7

[28] M.-Y. Shen and C.-C. Jay Kuo. Real-time compression artifact reduction via robust nonlinear filtering. In *ICIP*, volume 2, pages 565–569. IEEE, 1999. 1

[29] P. Sprechmann, A. Bronstein, and G. Sapiro. Learning efficient sparse and low rank models. *TPAMI*, 2015. 2, 4

[30] P. Sprechmann, R. Litman, T. B. Yakar, A. M. Bronstein, and G. Sapiro. Supervised sparse analysis and synthesis operators. In *NIPS*, pages 908–916, 2013. 4

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 4

[32] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 7

[33] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang. Learning a task-specific deep architecture for clustering. *SDM*, 2016. 2

[34] Z. Wang, Q. Ling, and T. Huang. Learning deep ℓ_0 encoders. *AAAI*, 2016. 2

[35] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. *ICCV*, 2015. 2, 4

[36] Z. Wang, Y. Yang, Z. Wang, S. Chang, W. Han, J. Yang, and T. Huang. Self-tuned deep super resolution. In *IEEE CVPR Workshops*, pages 1–8, 2015. 2

[37] C. Yim and A. C. Bovik. Quality assessment of deblocked images. *TIP*, 20(1):88–98, 2011. 7