Compression Artifacts Reduction by a Deep Convolutional Network

Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang
Department of Information Engineering, The Chinese University of Hong Kong
{dc012,dy015,ccloy,xtang}@ie.cuhk.edu.com

Abstract

*Lossy compression introduces complex compression artifacts, particularly the blocking artifacts, ringing effects and blurring. Existing algorithms either focus on removing blocking artifacts and produce blurred output, or restores sharpened images that are accompanied with ringing effects. Inspired by the deep convolutional networks (DCN) on super-resolution [5], we formulate a compact and efficient network for seamless attenuation of different compression artifacts. We also demonstrate that a deeper model can be effectively trained with the features learned in a shallow network. Following a similar "easy to hard" idea, we systematically investigate several practical transfer settings and show the effectiveness of transfer learning in low-level vision problems. Our method shows superior performance than the state-of-the-arts both on the benchmark datasets and the real-world use case (*i.e. *Twitter).*

1. Introduction

Lossy compression (*e.g.* JPEG, WebP and HEVC-MSP) is one class of data encoding methods that uses inexact approximations for representing the encoded content. In this age of information explosion, lossy compression is indispensable and inevitable for companies (*e.g.* *Twitter* and *Facebook*) to save bandwidth and storage space. However, compression in its nature will introduce undesired complex artifacts, which will severely reduce the user experience (*e.g.* Figure 1). All these artifacts not only decrease perceptual visual quality, but also adversely affect various low-level image processing routines that take compressed images as input, *e.g.* contrast enhancement [16], super-resolution [30, 5], and edge detection [3]. However, under such a huge demand, effective compression artifacts reduction remains an open problem.

We take JPEG compression as an example to explain compression artifacts. JPEG compression scheme divides an image into 8×8 pixel blocks and applies block discrete cosine transformation (DCT) on each block individually. Quantization is then applied on the DCT coefficients to save storage space. This step will cause a complex combination of different artifacts, as depicted in Figure 1(a). *Blocking artifacts* arise when each block is encoded without considering the correlation with the adjacent blocks, resulting in discontinuities at the 8×8 borders. *Ringing effects* along the edges occur due to the coarse quantization of the high-frequency components (also known as Gibbs phenomenon [9]). *Blurring* happens due to the loss of high-frequency components. To cope with the various compression artifacts, different approaches have been proposed, some of which can only deal with certain types of artifacts. For instance, deblocking oriented approaches [18, 21, 26] perform filtering along the block boundaries to reduce only blocking artifacts. Liew *et al.* [17] and Foi *et al.* [6] use thresholding by wavelet transform and Shape-Adaptive DCT transform, respectively. These approaches are good at

(a) Left: the JPEG-compressed image, where we could see blocking artifacts, ringing effects and blurring on the eyes, abrupt intensity changes on the face. Right: the restored image by the proposed deep model (AR-CNN), where we remove these compression artifacts and produce sharp details.

(b) Left: the Twitter-compressed image, which is first re-scaled to a small image and then compressed on the server-side. Right: the restored image by the proposed deep model (AR-CNN)

Figure 1. Example compressed images and our restoration results on the JPEG compression scheme and the real use case – *Twitter*.

removing blocking and ringing artifacts, but tend to produce blurred output. Jung *et al*. [13] propose restoration method based on sparse representation. They produce sharpened images but accompanied with noisy edges and unnatural smooth regions.

To date, deep learning has shown impressive results on both high-level and low-level vision problems . In particular, the SRCNN proposed by Dong *et al*. [5] shows the great potential of an end-to-end DCN in image super-resolution. The study also points out that conventional sparse-coding-based image restoration model can be equally seen as a deep model. However, we find that the three-layer network is not well suited in restoring the compressed images, especially in dealing with blocking artifacts and handling smooth regions. As various artifacts are coupled together, features extracted by the first layer is noisy, causing undesirable noisy patterns in reconstruction.

To eliminate the undesired artifacts, we improve the SRCNN by embedding one or more "feature enhancement" layers after the first layer to clean the noisy features. Experiments show that the improved model, namely "Artifacts Reduction Convolutional Neural Networks (AR-CNN)", is exceptionally effective in suppressing blocking artifacts while retaining edge patterns and sharp details (see Figure 1). However, we are met with training difficulties in training a deeper DCN. "Deeper is better" is widely observed in high-level vision problems, but not in low-level vision tasks. Specifically, "deeper is not better" has been pointed out in super-resolution [4], where training a five-layer network becomes a bottleneck. The difficulty of training is partially due to the sub-optimal initialization settings.

The aforementioned difficulty motivates us to investigate a better way to train a deeper model for low-level vision problems. We find that this can be effectively solved by transferring the features learned in a shallow network to a deeper one and fine-tuning simultaneously[1]. This strategy has also been proven successful in learning a deeper CNN for image classification [24]. Following a similar general intuitive idea, *easy to hard*, we discover other interesting transfer settings in this low-level vision task: (1) We transfer the features learned in a high-quality compression model (easier) to a low-quality one (harder), and find that it converges faster than random initialization. (2) In the real use case, companies tend to apply different compression strategies (including re-scaling) according to their purposes (*e.g*. Figure 1(b)). We transfer the features learned in a standard compression model (easier) to a real use case (harder), and find that it performs better than learning from scratch.

[1]Generally, the transfer learning method will train a base network first, and copy the learned parameters or features of several layers to the corresponding layers of a target network. These transferred layers can be left frozen or fine-tuned to the target dataset. The remaining layers are randomly initialized and trained to the target task.

The contributions of this study are two-fold: (1) We formulate a new deep convolutional network for efficient reduction of various compression artifacts. Extensive experiments, including that on real use cases, demonstrate the effectiveness of our method over state-of-the-art methods [6, 12] both perceptually and quantitatively. (2) We verify that reusing the features in shallow networks is helpful in learning a deeper model for compression artifact reduction. Under the same intuitive idea – *easy to hard*, we reveal a number of interesting and practical transfer settings. Our study is the first attempt to show the effectiveness of feature transfer in a low-level vision problem.

2. Related work

Existing algorithms can be classified into deblocking oriented and restoration oriented methods. The deblocking oriented methods focus on removing blocking and ringing artifacts. In the spatial domain, different kinds of filters [18, 21, 26] have been proposed to adaptively deal with blocking artifacts in specific regions (*e.g*., edge, texture, and smooth regions). In the frequency domain, Liew *et al*. [17] utilize wavelet transform and derive thresholds at different wavelet scales for denoising. The most successful deblocking oriented method is perhaps the Pointwise Shape-Adaptive DCT (SA-DCT) [6], which is widely acknowledged as the state-of-the-art approach [12, 16]. However, as most deblocking oriented methods, SA-DCT could not reproduce sharp edges, and tend to overly smooth texture regions. The restoration oriented methods regard the compression operation as distortion and propose restoration algorithms. They include projection on convex sets based method (POCS) [32], solving an MAP problem (FoE) [25], sparse-coding-based method [13] and the Regression Tree Fields based method (RTF) [12], which is the new state-of-the art method. The RTF takes the results of SA-DCT [6] as bases and produces globally consistent image reconstructions with a regression tree field model. It could also be optimized for any differentiable loss functions (*e.g*. SSIM), but often at the cost of other evaluation metrics.

Super-Resolution Convolutional Neural Network (SRCNN) [5] is closely related to our work. In the study, independent steps in the sparse-coding-based method are formulated as different convolutional layers and optimized in a unified network. It shows the potential of deep model in low-level vision problems like super-resolution. However, the model of compression is different from super-resolution in that it consists of different kinds of artifacts. Designing a deep model for compression restoration requires a deep understanding into the different artifacts. We show that directly applying the SRCNN architecture for compression restoration will result in undesired noisy patterns in the reconstructed image.

Transfer learning in deep neural networks becomes pop-

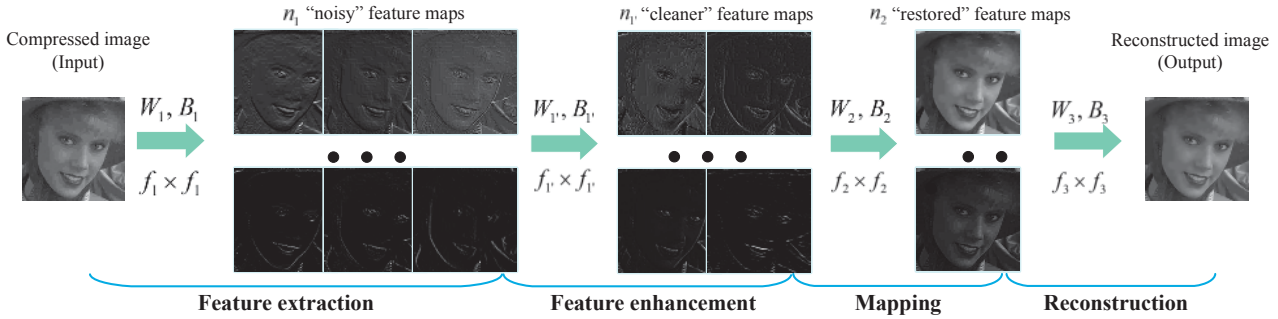


Figure 2. The framework of the Artifacts Reduction Convolutional Neural Network (AR-CNN). The network consists of four convolutional layers, each of which is responsible for a specific operation. Then it optimizes the four operations (*i.e.*, feature extraction, feature enhancement, mapping and reconstruction) jointly in an end-to-end framework. Example feature maps shown in each step could well illustrate the functionality of each operation. They are normalized for better visualization.

ular since the success of deep learning in image classification [15]. The features learned from the ImageNet show good generalization ability [35] and become a powerful tool for several high-level vision problems, such as Pascal VOC image classification [20] and object detection [7, 22]. Yosinski *et al.* [34] have also tried to quantify the degree to which a particular layer is general or specific. Overall, transfer learning has been systematically investigated in high-level vision problems, but not in low-level vision tasks. In this study, we explore several transfer settings on compression artifacts reduction and show the effectiveness of transfer learning in low-level vision problems.

3. Methodology

Our proposed approach is based on the current successful low-level vision model – SRCNN [5]. To have a better understanding of our work, we first give a brief overview of SRCNN. Then we explain the insights that lead to a deeper network and present our new model. Subsequently, we explore three types of transfer learning strategies that help in training a deeper and better network.

3.1. Review of SRCNN

The SRCNN aims at learning an end-to-end mapping, which takes the low-resolution image $\mathbf{Y}$ (after interpolation) as input and directly outputs the high-resolution one $F(\mathbf{Y})$. The network contains three convolutional layers, each of which is responsible for a specific task. Specifically, the first layer performs **patch extraction and representation**, which extracts overlapping patches from the input image and represents each patch as a high-dimensional vector. Then the **non-linear mapping** layer maps each high-dimensional vector of the first layer to another high-dimensional vector, which is conceptually the representation of a high-resolution patch. At last, the **reconstruction** layer aggregates the patch-wise representations to generate the final output. The network can be expressed as:

$$F_i(\mathbf{Y}) = \max\left(0, W_i * \mathbf{Y} + B_i\right), i \in \{1, 2\}; \quad (1)$$

$$F(\mathbf{Y}) = W_3 * F_2(\mathbf{Y}) + B_3. \quad (2)$$

where W_i and B_i represent the filters and biases of the ith layer respectively, F_i is the output feature maps and '$*$' denotes the convolution operation. The W_i contains n_i filters of support $n_{i-1} \times f_i \times f_i$, where f_i is the spatial support of a filter, n_i is the number of filters, and n_0 is the number of channels in the input image. Note that there is no pooling or full-connected layers in SRCNN, so the final output $F(\mathbf{Y})$ is of the same size as the input image. Rectified Linear Unit (ReLU, $\max(0, x)$) [19] is applied on the filter responses.

These three steps are analogous to the basic operations in the sparse-coding-based super-resolution methods [31], and this close relationship lays theoretical foundation for its successful application in super-resolution. Details can be found in the paper [5].

3.2. Convolutional Neural Network for Compression Artifacts Reduction

Insights. In sparse-coding-based methods and SRCNN, the first step – feature extraction – determines what should be emphasized and restored in the following stages. However, as various compression artifacts are coupled together, the extracted features are usually noisy and ambiguous for accurate mapping. In the experiments of reducing JPEG compression artifacts (see Section 4.1.2), we find that some quantization noises coupled with high frequency details are further enhanced, bringing unexpected noisy patterns around sharp edges. Moreover, blocking artifacts in flat areas are misrecognized as normal edges, causing abrupt intensity changes in smooth regions. Inspired by the feature enhancement step in super-resolution [29], we introduce a feature enhancement layer after the feature extraction layer in SRCNN to form a new and deeper network

– AR-CNN. This layer maps the "noisy" features to a relatively "cleaner" feature space, which is equivalent to denoising the feature maps.

Formulation. The overview of the new network AR-CNN is shown in Figure 2. The three layers of SRCNN remain unchanged in the new model. We also use the same annotations as in Section 3.1. To conduct feature enhancement, we extract new features from the n_1 feature maps of the first layer, and combine them to form another set of feature maps. This operation $F_{1'}$ can also be formulated as a convolutional layer:

$$F_{1'}(\mathbf{Y}) = \max\left(0, W_{1'} * F_1(\mathbf{Y}) + B_{1'}\right), \qquad (3)$$

where $W_{1'}$ corresponds to $n_{1'}$ filters with size $n_1 \times f_{1'} \times f_{1'}$. $B_{1'}$ is an $n_{1'}$-dimensional bias vector, and the output $F_{1'}(\mathbf{Y})$ consists of $n_{1'}$ feature maps. Overall, the AR-CNN consists of four layers, namely the feature extraction, feature enhancement, mapping and reconstruction layer.

It is worth noticing that AR-CNN is not equal to a deeper SRCNN that contains more than one non-linear mapping layers[2]. A deeper SRCNN imposes more non-linearity in the mapping stage, which equals to adopting a more robust regressor between the low-level features and the final output. Similar ideas have been proposed in some sparse-coding-based methods [14, 2]. However, as the compression artifacts are complex, low-level features extracted by a single layer are noisy. Thus the performance bottleneck lies on the features but not the regressor. AR-CNN improves the mapping accuracy by enhancing the extracted low-level features, and the first two layers together can be regarded as a better feature extractor. This leads to better performance than a deeper SRCNN. Experimental results of AR-CNN, SRCNN and deeper SRCNN will be shown in Section 4.1.2.

3.3. Model Learning

Given a set of ground truth images $\{\mathbf{X}_i\}$ and their corresponding compressed images $\{\mathbf{Y}_i\}$, we use Mean Squared Error (MSE) as the loss function:

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i; \Theta) - \mathbf{X}_i||^2, \qquad (4)$$

where $\Theta = \{W_1, W_{1'}, W_2, W_3, B_1, B_{1'}, B_2, B_3\}$, n is the number of training samples. The loss is minimized using stochastic gradient descent with the standard backpropagation. We adopt a batch-mode learning method with a batch size of 128.

3.4. Easy-Hard Transfer

Transfer learning in deep models provides an effective way of initialization. In fact, conventional initialization strategies (*i.e.* randomly drawn from Gaussian distributions with fixed standard deviations [15]) are found not suitable for training a very deep model, as reported in [10]. To address this issue, He *et al*. [10] derive a robust initialization method for rectifier nonlinearities, Simonyan *et al*. [24] propose to use the pre-trained features on a shallow network for initialization.

In low-level vision problems (*e.g.* super resolution), it is observed that training a network beyond 4 layers would encounter the problem of convergence, even that a large number of training images (*e.g.* ImageNet) are provided [5]. We are also met with this difficulty during the training process of AR-CNN. To this end, we systematically investigate several transfer settings in training a low-level vision network following an intuitive idea of "easy-hard transfer". Specifically, we attempt to reuse the features learned in a relatively easier task to initialize a deeper or harder network. Interestingly, the concept "easy-hard transfer" has already been pointed out in neuro-computation study [8], where the prior training on an easy discrimination can help learn a second harder one.

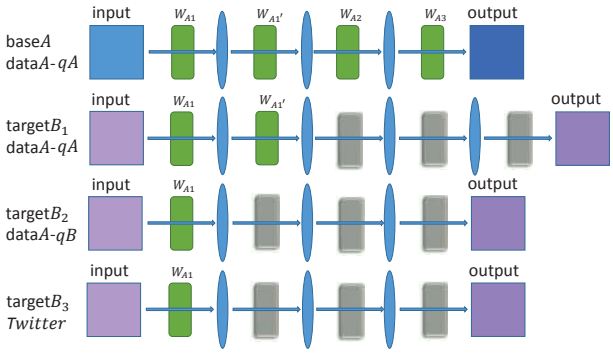
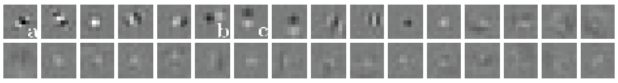


Figure 3. Easy-hard transfer settings. First row: The baseline 4-layer network trained with *data*A-qA. Second row: The 5-layer AR-CNN targeted at *data*A-qA. Third row: The AR-CNN targeted at *data*A-qB. Fourth row: The AR-CNN targeted at *Twitter* data. Green boxes indicate the transferred features from the base network, and gray boxes represent random initialization. The ellipsoidal bars between weight vectors represent the activation functions.

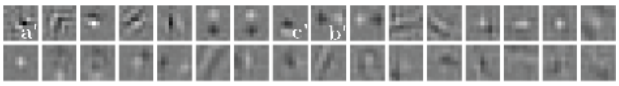
Formally, we define the base (or source) task as A and the target tasks as B_i, $i \in \{1, 2, 3\}$. As shown in Figure 3, the base network *base*A is a four-layer AR-CNN trained on a large dataset *data*A, of which images are compressed using a standard compression scheme with the compression quality qA. All layers in *base*A are randomly initialized from a Gaussian distribution. We will transfer one or two layers of *base*A to different target tasks (see Figure 3). Such transfers can be described as follows.

Transfer shallow to deeper model. As indicated by [4], a five-layer network is sensitive to the initialization parameters and learning rate. Thus we transfer the first two layers of *base*A to a five-layer network *target*B_1. Then we ran-

[2]Adding non-linear mapping layers has been suggested as an extension of SRCNN in [5].



(a) High compression quality (quality 20 in Matlab encoder)



(b) Low compression quality (quality 10 in Matlab encoder)

Figure 4. First layer filters of AR-CNN learned under different JPEG compression qualities.

domly initialize its remaining layers[3] and train all layers toward the same dataset *dataA*. This is conceptually similar to that applied in image classification [24], but this approach has never been validated in low-level vision problems.

Transfer high to low quality. Images of low compression quality contain more complex artifacts. Here we use the features learned from high compression quality images as a starting point to help learn more complicated features in the DCN. Specifically, the first layer of *target*B_2 are copied from *baseA* and trained on images that are compressed with a lower compression quality *qB*.

Transfer standard to real use case. We then explore whether the features learned under a standard compression scheme can be generalized to other real use cases, which often contain more complex artifacts due to different levels of re-scaling and compression. We transfer the first layer of *baseA* to the network *target*B_3, and train all layers on the new dataset.

Discussion. Why the features learned from relatively easy tasks are helpful? First, the features from a well-trained network can provide a good starting point. Then the rest of a deeper model can be regarded as shallow one, which is easier to converge. Second, features learned in different tasks always have a lot in common. For instance, Figure 3.4 shows the features learned under different JPEG compression qualities. Obviously, filters a, b, c of high quality are very similar to filters a', b', c' of low quality. This kind of features can be reused or improved during fine-tuning, making the convergence faster and more stable. Furthermore, a deep network for a hard problem can be seen as an insufficiently biased learner with overly large hypothesis space to search, and therefore is prone to overfitting. These few transfer settings we investigate introduce good bias to enable the learner to acquire a concept with greater generality. Experimental results in Section 4.2 validate the above analysis.

[3]Random initialization on remaining layers are also applied similarly for tasks B_2, and B_3.

4. Experiments

We use the BSDS500 database [1] as our base training set. Specifically, its disjoint training set (200 images) and test set (200 images) are all used for training, and its validation set (100 images) is used for validation. As in other compression artifacts reduction methods (*e.g.* RTF [12]), we apply the standard JPEG compression scheme, and use the JPEG quality settings $q = 40, 30, 20, 10$ (from high quality to very low quality) in MATLAB JPEG encoder. We only focus on the restoration of the luminance channel (in YCrCb space) in this paper.

The training image pairs $\{Y, X\}$ are prepared as follows – Images in the training set are decomposed into 32×32 sub-images[4] $X = \{\mathbf{X}_i\}_{i=1}^n$. Then the compressed samples $Y = \{\mathbf{Y}_i\}_{i=1}^n$ are generated from the training samples with MATLAB JPEG encoder [12]. The sub-images are extracted from the ground truth images with a stride of 10. Thus the 400 training images could provide 537,600 training samples. To avoid the border effects caused by convolution, AR-CNN produces a 20×20 output given a 32×32 input $\mathbf{Y}_i$. Hence, the loss (Eqn. (4)) was computed by comparing against the center 20×20 pixels of the ground truth sub-image $\mathbf{X}_i$. In the training phase, we follow [11, 5] and use a smaller learning rate (10^{-5}) in the last layer and a comparably larger one (10^{-4}) in the remaining layers.

4.1. Comparison with the State-of-the-Arts

We use the LIVE1 dataset [23] (29 images) as test set to evaluate both the quantitative and qualitative performance. The LIVE1 dataset contains images with diverse properties. It is widely used in image quality assessment [27] as well as in super-resolution [30]. To have a comprehensive qualitative evaluation, we apply the PSNR, structural similarity (SSIM) [27][5], and PSNR-B [33] for quality assessment. We want to emphasize the use of PSNR-B. It is designed specifically to assess blocky and deblocked images. The network settings are $f_1 = 9$, $f_{1'} = 7$, $f_2 = 1$, $f_3 = 5$, $n_1 = 64$, $n_{1'} = 32$, $n_2 = 16$ and $n_3 = 1$, denoted as AR-CNN (9-7-1-5) or simply AR-CNN. A specific network is trained for each JPEG quality. Parameters are randomly initialized from a Gaussian distribution with a standard deviation of 0.001.

4.1.1 Comparison with SA-DCT

We first compare AR-CNN with SA-DCT [6], which is widely regarded as the state-of-the-art deblocking oriented method [12, 16]. The quantization results of PSNR, SSIM

[4]We use sub-images because we regard each sample as an image rather than a big patch.

[5]We use the unweighted structural similarity defined over fixed 8×8 windows as in [28].

Table 1. The average results of PSNR (dB), SSIM, PSNR-B (dB) on the LIVE1 dataset.

Eval. Mat	Quality	JPEG	SA-DCT	AR-CNN
PSNR	10	27.77	28.65	**28.98**
	20	30.07	30.81	**31.29**
	30	31.41	32.08	**32.69**
	40	32.35	32.99	**33.63**
SSIM	10	0.7905	0.8093	**0.8217**
	20	0.8683	0.8781	**0.8871**
	30	0.9000	0.9078	**0.9166**
	40	0.9173	0.9240	**0.9306**
PSNR-B	10	25.33	28.01	**28.70**
	20	27.57	29.82	**30.76**
	30	28.92	30.92	**32.15**
	40	29.96	31.79	**33.12**

Table 2. The average results of PSNR (dB), SSIM, PSNR-B (dB) on 5 classical test images [6].

Eval. Mat	Quality	JPEG	SA-DCT	AR-CNN
PSNR	10	27.82	28.88	**29.04**
	20	30.12	30.92	**31.16**
	30	31.48	32.14	**32.52**
	40	32.43	33.00	**33.34**
SSIM	10	0.7800	0.8071	**0.8111**
	20	0.8541	0.8663	**0.8694**
	30	0.8844	0.8914	**0.8967**
	40	0.9011	0.9055	**0.9101**
PSNR-B	10	25.21	28.16	**28.75**
	20	27.50	29.75	**30.60**
	30	28.94	30.83	**31.99**
	40	29.92	31.59	**32.80**

and PSNR-B are shown in Table 1. On the whole, our AR-CNN outperforms the SA-DCT on all JPEG qualities and evaluation metrics by a large margin. Note that the gains on PSNR-B is much larger than that on PSNR. This indicates that AR-CNN could produce images with less blocking artifacts. We have also conducted evaluation on 5 classical test images used in [6][6], and observed the same trend. The results are shown in Table 2.

To compare the visual quality, we present some restored images[7] with $q = 10$ in Figure 10. From Figure 10, we could see that the result of AR-CNN could produce much sharper edges with much less blocking and ringing artifacts compared with SA-DCT. The visual quality has been largely improved on all aspects compared with the state-of-the-art method. Furthermore, AR-CNN is superior to SA-DCT on the implementation speed. For SA-DCT, it needs 3.4 seconds to process a 256×256 image. While AR-CNN only takes 0.5 second. They are all implemented using C++ on a PC with Intel I3 CPU (3.1GHz) with 16GB RAM.

4.1.2 Comparison with SRCNN

As discussed in Section 3.2, SRCNN is not suitable for compression artifacts reduction. For comparison, we train two SRCNN networks with different settings. (i) The original SRCNN (9-1-5) with $f_1 = 9$, $f_3 = 5$, $n_1 = 64$ and $n_2 = 32$. (ii) Deeper SRCNN (9-1-1-5) with an additional non-linear mapping layer ($f_{2'} = 1$, $n_{2'} = 16$). They all use the BSDS500 dataset for training and validation as in Section 4. The compression quality is $q = 10$. The AR-CNN is the same as in Section 4.1.1.

[6]The 5 test images in [6] are baboon, barbara, boats, lenna and peppers.

[7]More qualitative results are provided in the supplementary file.

Table 3. The average results of PSNR (dB), SSIM, PSNR-B (dB) on the LIVE1 dataset with $q = 10$.

Eval. Mat	JPEG	SRCNN	Deeper SRCNN	AR-CNN
PSNR	27.77	28.91	28.92	**28.98**
SSIM	0.7905	0.8175	0.8189	**0.8217**
PSNR-B	25.33	28.52	28.46	**28.70**

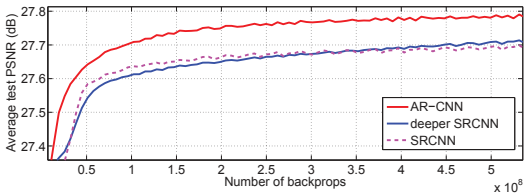


Figure 5. Comparisons with SRCNN and Deeper SRCNN.

Table 4. The average results of PSNR (dB), SSIM, PSNR-B (dB) on the test set BSDS500 dataset.

Eval. Mat	Quality	JPEG	RTF	RTF +SA-DCT	AR-CNN
PSNR	10	26.62	27.66	27.71	**27.79**
	20	28.80	29.84	29.87	**30.00**
SSIM	10	0.7904	0.8177	0.8186	**0.8228**
	20	0.8690	0.8864	0.8871	**0.8899**
PSNR-B	10	23.54	26.93	26.99	**27.32**
	20	25.62	28.80	28.80	**29.15**

Quantitative results tested on LIVE1 dataset are shown in Table 3. We could see that the two SRCNN networks are inferior on all evaluation metrics. From convergence curves shown in Figure 5, it is clear that AR-CNN achieves higher PSNR from the beginning of the learning stage. Furthermore, from their restored images[7] in Figure 11, we find out that the two SRCNN networks all produce images with noisy edges and unnatural smooth regions. These results demonstrate our statements in Section 3.2. In short, the success of training a deep model needs comprehensive understanding of the problem and careful design of the model structure.

4.1.3 Comparison with RTF

RTF [12] is the recent state-of-the-art restoration oriented method. Without their deblocking code, we can only compare with the released deblocking results. Their model is trained on the training set (200 images) of the BSDS500 dataset, but all images are down-scaled by a factor of 0.5 [12]. To have a fair comparison, we also train new AR-CNN networks on the same half-sized 200 images. Test-

Table 5. Experimental settings of "easy-hard transfer".

transfer strategy	short form	network structure	training dataset	initialization strategy
base network	base-q10	9-7-1-5	BSDS-q10	Gaussian (0, 0.001)
	base-q20	9-7-1-5	BSDS-q20	Gaussian (0, 0.001)
shallow to deep	base-q10	9-7-1-5	BSDS-q10	Gaussian (0, 0.001)
	transfer deeper	9-7-3-1-5	BSDS-q10	1,2 layers of base-q10
	He [10]	9-7-3-1-5	BSDS-q10	He *et al.* [10]
high to low	base-q10	9-7-1-5	BSDS-q10	Gaussian (0, 0.001)
	transfer 1 layer	9-7-1-5	BSDS-q10	1 layer of base-q20
	transfer 2 layers	9-7-1-5	BSDS-q10	1,2 layer of base-q20
standard to real	base-Twitter	9-7-1-5	*Twitter*	Gaussian (0, 0.001)
	transfer q10	9-7-1-5	*Twitter*	1 layer of base-q10
	transfer q20	9-7-1-5	*Twitter*	1 layer of base-q20

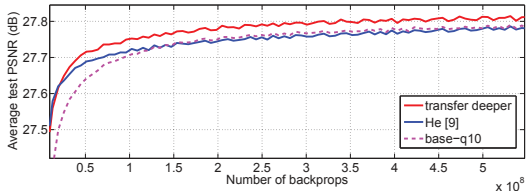


Figure 6. Transfer shallow to deeper model.

ing is performed on the test set of the BSDS500 dataset (images scaled by a factor of 0.5), which is also consistent with [12]. We compare with two RTF variants. One is the plain RTF, which uses the filter bank and is optimized for PSNR. The other is the RTF+SA-DCT, which includes the SA-DCT as a base method and is optimized for MAE. The later one achieves the highest PSNR value among all RTF variants [12].

As shown in Table 4, we obtain superior performance than the plain RTF, and even better performance than the combination of RTF and SA-DCT, especially under the more representative PSNR-B metric. Moreover, training on such a small dataset has largely restricted the ability of AR-CNN. The performance of AR-CNN will further improve given more training images.

4.2. Experiments on Easy-Hard Transfer

We show the experimental results of different "easy-hard transfer" settings, of which the details are shown in Table 5. Take the base network as an example, the base-q10 is a four-layer AR-CNN (9-7-1-5) trained on the BSDS500 [1] dataset (400 images) under the compression quality $q = 10$. Parameters are initialized by randomly drawing from a Gaussian distribution with zero mean and standard deviation 0.001. Figures 6 - 8 show the convergence curves on the validation set.

4.2.1 Transfer shallow to deeper model

In Table 5, we denote a deeper (five-layer) AR-CNN as "9-7-3-1-5", which contains another feature enhancement layer ($f_{1''} = 3$ and $n_{1''} = 16$). Results in Figure 6 show that the

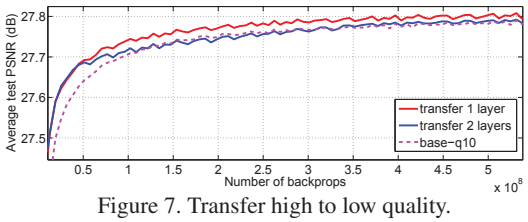


Figure 7. Transfer high to low quality.

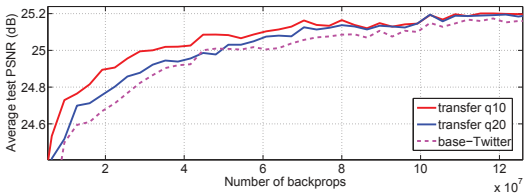


Figure 8. Transfer standard to real use case.

transferred features from a four-layer network enable us to train a five-layer network successfully. Note that directly training a five-layer network using conventional initialization ways is unreliable. Specifically, we have exhaustively tried different groups of learning rates, but still have not observed convergence. Furthermore, the "transfer deeper" converges faster and achieves better performance than using He *et al.*'s method [10], which is also very effective in training a deep model. We have also conducted comparative experiments with the structure "9-7-1-1-5" and observed the same trend.

4.2.2 Transfer high to low quality

Results are shown in Figure 7. Obviously, the two networks with transferred features converge faster than that training from scratch. For example, to reach an average PSNR of 27.77dB, the "transfer 1 layer" takes only 1.54×10^8 backprops, which are roughly a half of that for "base-q10". Moreover, the "transfer 1 layer" also outperforms the 'base-q10" by a slight margin throughout the training phase. One reason for this is that only initializing the first layer provides the network with more flexibility in adapting to a new dataset. This also indicates that a good starting point could help train a better network with higher convergence speed.

4.2.3 Transfer standard to real use case – *Twitter*

Online Social Media like *Twitter* are popular platforms for message posting. However, *Twitter* will compress the uploaded images on the server-side. For instance, a typical 8 mega-pixel (MP) image (3264×2448) will result in a compressed and re-scaled version with a fixed resolution of 600×450. Such re-scaling and compression will introduce very complex artifacts, making restoration difficult for existing deblocking algorithms (*e.g.* SA-DCT). However, AR-CNN can fit to the new data easily. Further, we want

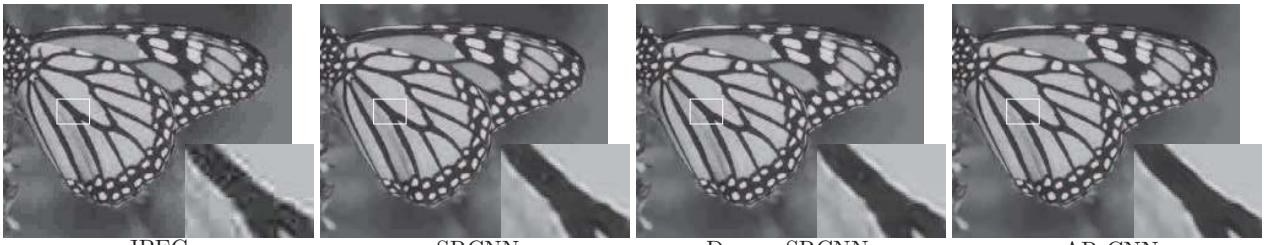
Original
PSNR /SSIM /PSNR-B

JPEG
32.46 dB /0.8558 /29.64 dB

SA-DCT
33.88 dB /0.9015 /33.02 dB

AR-CNN
34.37 dB /**0.9079** /**34.10** dB

Figure 10. Results on image "parrots" show that AR-CNN is better than SA-DCT on removing blocking artifacts.



JPEG
30.12 dB /0.8817 /26.86 dB

SRCNN
32.58 dB /0.9298 /31.52 dB

Deeper SRCNN
32.60 dB /0.9301 /31.47 dB

AR-CNN
32.88 dB /**0.9343** /**32.22** dB

Figure 11. Results on image "monarch" show that AR-CNN is better than SRCNN on removing ringing effects.

Original / PSNR

Twitter / 26.55 dB

Base-*Twitter* / 27.75 dB

Transfer q10 / **27.92** dB

Figure 12. Restoration results of AR-CNN on *Twitter* compressed images. The origina image (8MP version) is too large for display and only part of the image is shown for better visualization.

to show that features learned under standard compression schemes could also facilitate training on a completely different dataset. We use 40 photos of resolution 3264×2448 taken by mobile phones (totally 335,209 training subimages) and their *Twitter*-compressed version[8] to train three networks with initialization settings listed in Table 5.

From Figure 8, we observe that the "transfer $q10$" and "transfer $q20$" networks converge much faster than the "base-Twitter" trained from scratch. Specifically, the "transfer $q10$" takes 6×10^7 backprops to achieve 25.1dB, while the "base-Twitter" uses 10×10^7 backprops. Despite of fast convergence, transferred features also lead to higher PSNR values compared with "base-Twitter". This observation suggests that features learned under standard compression schemes are also transferrable to tackle real use case problems. Some restoration results[7] are shown in Figure 12. We could see that both networks achieve satisfactory quality improvements over the compressed version.

5. Conclusion

Applying deep model on low-level vision problems requires deep understanding of the problem itself. In this paper, we carefully study the compression process and propose a four-layer convolutional network, AR-CNN, which is extremely effective in dealing with various compression artifacts. We further systematically investigate several *easy-to-hard* transfer settings that could facilitate training a deeper or better network, and verify the effectiveness of transfer learning in low-level vision problems. As discussed in SRCNN [5], we find that larger filter sizes also help improve the performance. We will leave them to further work.

[8] We will share this dataset on our project page.

References

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

[2] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[3] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, pages 1841–1848. IEEE, 2013.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *arXiv:1501.00092*, 2014.

[5] 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.

[6] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images. *TIP*, 16(5):1395–1411, 2007.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587. IEEE, 2014.

[8] M. A. Gluck and C. E. Myers. Hippocampal mediation of stimulus representation: A computational theory. *Hippocampus*, 3(4):491–516, 1993.

[9] R. C. Gonzalez and R. E. Woods. Digital image processing, 2002.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv:1502.01852*, 2015.

[11] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *NIPS*, pages 769–776, 2009.

[12] 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. 2012.

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

[14] K. I. Kim and Y. Kwon. Single-image super-resolution using sparse regression and natural image prior. 32(6):1127–1133, 2010.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[16] Y. Li, F. Guo, R. T. Tan, and M. S. Brown. A contrast enhancement framework with jpeg artifacts suppression. In *ECCV*, pages 174–188. 2014.

[17] A.-C. Liew and H. Yan. Blocking artifacts suppression in block-coded images using overcomplete wavelet representation. *TCSVT*, 14(4):450–461, 2004.

[18] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *TCSVT*, 13(7):614–619, 2003.

[19] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, pages 807–814, 2010.

[20] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724. IEEE, 2014.

[21] H. C. Reeve III and J. S. Lim. Reduction of blocking effects in image coding. *Optical Engineering*, 23(1):230134–230134, 1984.

[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv:1312.6229*, 2013.

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

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[25] D. Sun and W.-K. Cham. Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior. *TIP*, 16(11):2743–2751, 2007.

[26] C. Wang, J. Zhou, and S. Liu. Adaptive non-local means filter for image deblocking. *Signal Processing: Image Communication*, 28(5):522–530, 2013.

[27] 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.

[28] Z. Wang and E. P. Simoncelli. Maximum differentiation (MAD) competition: A methodology for comparing computational models of perceptual quantities. *Journal of Vision*, 8(12):8, 2008.

[29] Z. Xiong, X. Sun, and F. Wu. Image hallucination with feature enhancement. In *CVPR*, pages 2074–2081. IEEE, 2009.

[30] C.-Y. Yang, C. Ma, and M.-H. Yang. Single-image super-resolution: A benchmark. In *ECCV*, pages 372–386. 2014.

[31] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. 19(11):2861–2873, 2010.

[32] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos. Projection-based spatially adaptive reconstruction of block-transform compressed images. *TIP*, 4(7):896–908, 1995.

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

[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[35] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. 2014.