COMPRESSION ARTIFACT SUPPRESSION FOR COLOR IMAGES WITH DUAL-DOMAIN SE-ARRESNET

BY

JIAXI JASON NIE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Minh N. Do

ABSTRACT

JPEG compression has been a popular lossy image compression technique and is widely used in digital imaging. Restoring high-quality images from their compressed JPEG counterparts, however, is an ill-posed inverse problem but could be of great use in improving the visual quality of images. With the representational power that convolutional neural networks (CNNs) demonstrate, we show that it is possible to suppress JPEG compression artifacts and recover visually pleasing images.

To recover original high-quality and high-resolution images from JPEG compressed images, we leverage prior knowledge of JPEG compression into consideration by exploiting frequency redundancies with the CNN in discrete cosine domain and constrain the quantization loss, in addition to exploiting spatial redundancies in the pixel domain. This data-driven approach targets removing compression artifacts, including blocking, blurring, ringing and banding artifacts, and recovering high-frequency information for reconstruction. We design a deep CNN in each domain and fuse the outputs with an aggregation network to produce the output image. To improve the model performance, we leverage the robustness and ability to tackle vanishing gradient problems of ResNet to build a deep network, and utilize squeeze-and-excitation block, a technique typically found beneficial in classification tasks, to this regression problem to exploit global information in a larger scale. We refer to the module proposed in this work as squeeze-and-excitation artifact

removal ResNet (SE-ARResNet). Prior work in this field mainly focuses on reconstructing a grayscale image or the luminance channel of the image. We demonstrate that we can reconstruct color images effectively and robustly with the dual-domain CNN approach.

To my parents, for their unconditional love and support.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my adviser, Professor Minh N. Do, for his continuous help and guidance throughout the development of this project. I also wish to extend my gratitude to Cu Khoi N. Mac, Raymond A. Yeh, Renan A. R. Gomez and Dr. Chen Chen for their help and constructive critiques. I gratefully acknowledge the support of NVIDIA Corporation for its donation of the Titan V GPU and Dr. Chen Chen for his donation of the GeForce RTX 2080Ti GPU used in this research. Last but not least I'd like to thank my parents, for their long-lasting love, support and sacrifice.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

With the rapid development of the Internet, digital imaging and mobile devices, transferring and displaying high-quality images has become ubiquitous. The growing demand for high-quality visually pleasing images requires high-resolution images with minimal noise and artifacts to be displayed on high-resolution screens, which is unfortunately limited by the constraints in network bandwidth and the amount of storage resource as the communication and storing are typically expensive. To meet this demand while coping with the aforementioned bottlenecks, several image compression techniques have been developed and widely applied over the past a few decades which generate highly compressed images with low bitrate at the cost of almost unnoticeable degradation.

In order to meet the bit-budget, many compression techniques, especially lossy ones (JPEG, JPEG 2000, WebP, HEVC-MSP, etc.), have been adopted to drop certain information during the compression process by taking advantage of the different sensitivities of human eyes. This process introduces irreversible loss in the low-bitrate image signal and poses a challenge in reconstructing the original high-quality image. JPEG compression was released in 1992 by the Joint Photographic Experts Group and is still one of the most popular compression techniques used today because of its high compression ratio and small perceptible loss. These quality degradations were typically small and went unnoticeable at the time of release of JPEG, but with the

development of high-resolution displays they can be easily spotted by human eyes, and are therefore unacceptable.

One way to tackle this problem is through progress in communication and storage technologies to efficiently transfer and store high-quality original images. Another way is through effective restoration techniques that can reconstruct original images from their compressed counterparts.

Two representative types of the latter approach have been shown useful in tackling this challenge and correcting the corrupted images: sparsity coding based approaches and convolutional neural network (CNN) approaches. The CNN approach has been demonstrated effective in reconstructing grayscale images or the luminance channel of color images. Theoretically these networks can be adopted to restore color images, but practically the result is not guaranteed to be as satisfying as that of a single-channel image. We show that recovering color images poses additional challenges compared to recovering single-channel images, and we propose a robust and effective framework to reconstruct original color images by suppressing compression artifacts and recovering high-frequency information. Our proposed model incorporates state-of-the-art techniques to fully exploit the power of the CNN and demonstrates satisfying performance in correcting compression artifacts.

1.1 Image Compression

Image compression is a major technique in computer vision to reduce the bitrate of high-resolution images with relatively large file sizes. These compression techniques output images with small file sizes such that they can be easily transmitted or stored using fewer bits than the original images to save precious transmission and storage resources. This compression is achieved

by exploiting correlations between pixel intensities and color components. Various compression techniques have been proposed, of which there are two distinct approaches: lossless and lossy. Lossless compression (PNG, GIF, etc.) is the compression that enables recovery of the original image, while lossy compression (JPEG, GIF, etc.) precludes recovery as it drops bits that do not significantly contribute to the resulting image quality. Compared to lossless compression, lossy compression can further reduce the bits required, and this reduction can be measured with the compression ratio defined by:

$$r = \frac{N_{original}(bits)}{N_{compressed}(bits)} \tag{1.1}$$

Because of its higher compression ratio, we focus on lossy compression, specifically JPEG compression, as lossless compressions can fully reconstruct the original images but are less popular.

1.2 JPEG Compression

The JPEG compression standard was released by the Joint Photographic Experts Group in 1992 to define and standardize a codec process for compressing images into bit streams and decoding back into pixels for storage and transmission. JPEG is a widely used transform coding technique and by transforming the image into frequency domain, it exploits the correlations to eliminate less-frequent information that is not noticeable in the resulting images. The compression quality is controlled by the parameter Quality Factor (QF), which is an integer between 0 and 100. Compression with a lower QF produces an image with smaller file size but more perceptible loss (see Fig. 1.1).

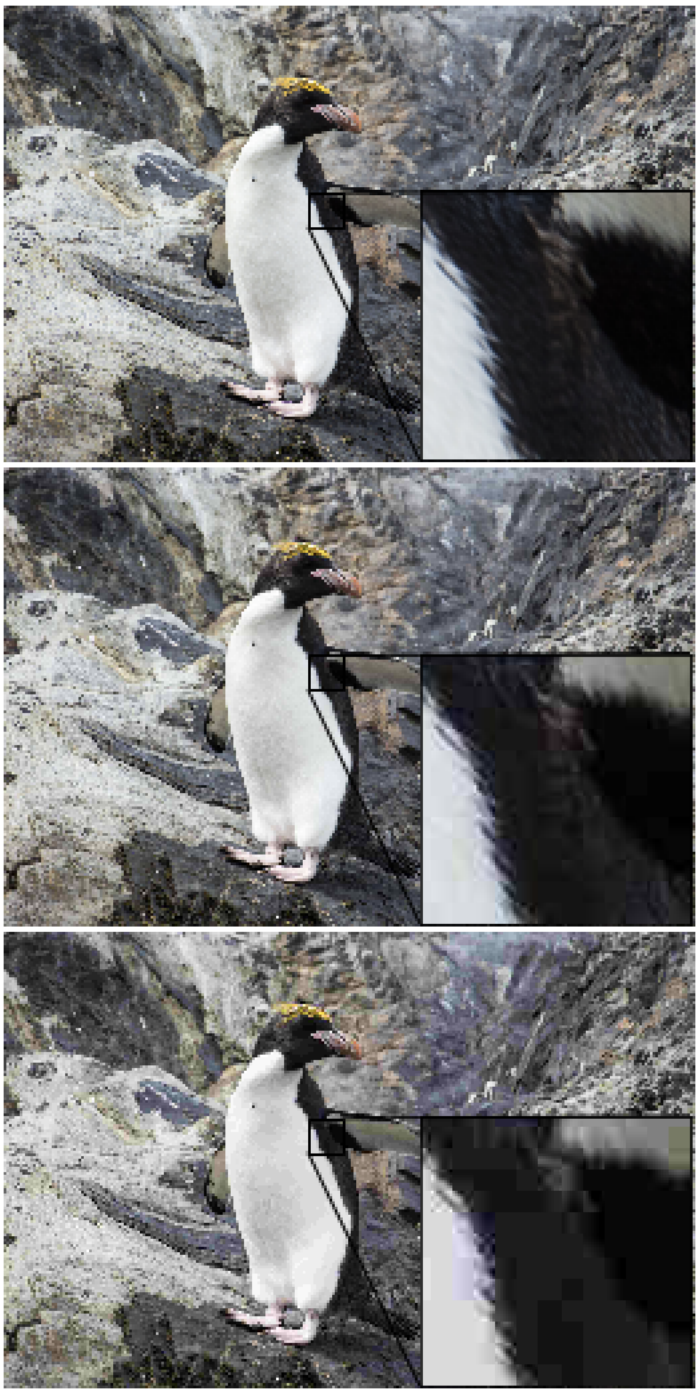

Figure 1.1: Top: Original high-resolution image. Middle: Image compressed with QF 30. Bottom: Image compressed with QF 10. Best viewed in color.

The detailed encoding process is diagramed in Fig. 1.2 and explained in the following section. For simplicity we denote the original image as $\boldsymbol{x}$ and

the compressed image as $\hat{\boldsymbol{x}}$ in the rest of this thesis.

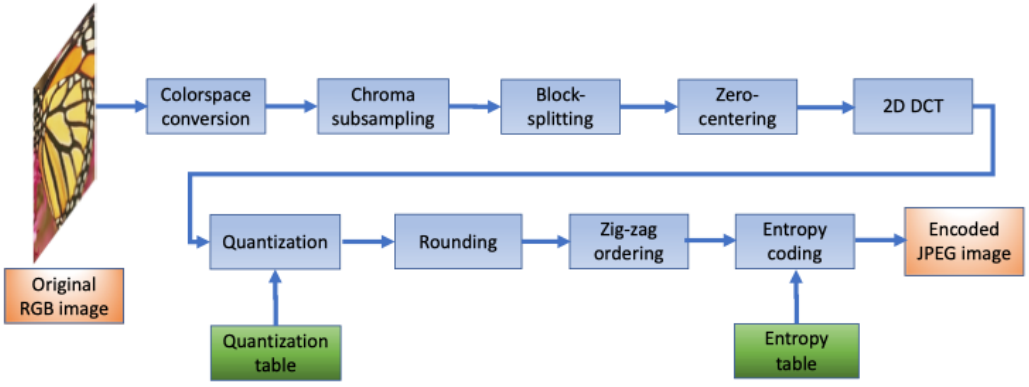

Figure 1.2: JPEG encoding pipeline.

1.2.1 Colorspace Transformation

The first step of the JPEG compression pipeline is to transform images from RGB colorspace into YCbCr colorspace by:

$$\begin{cases} \boldsymbol{x_Y} = 0 + 0.299 \cdot \boldsymbol{x_R} + 0.587 \cdot \boldsymbol{x_G} + 0.114 \cdot \boldsymbol{x_B} \\ \boldsymbol{x_{C_b}} = 128 - 0.169 \cdot \boldsymbol{x_R} - 0.331 \cdot \boldsymbol{x_G} + 0.5 \cdot \boldsymbol{x_B} \\ \boldsymbol{x_{C_r}} = 128 + 0.5 \cdot \boldsymbol{x_R} - 0.419 \cdot \boldsymbol{x_G} - 0.081 \cdot \boldsymbol{x_B} \end{cases} \tag{1.2}$$

with three decimal points of precision. It is worth noting that in practice images are typically represented by 8 bit unsigned integers or 32/64 bit floating numbers, and the level of precision introduces negligible error in colorspace conversion and final reconstruction. This transformation to YCbCr colorspace is desired because the primary colors in RGB colorspace are highly correlated, while channels in YCbCr colorspace eliminate this redundancy and better approximate perceptual uniformity. Specifically, the luminance channel (Y) captures the brightness of a pixel, while the chrominance channels (C_b and C_r) represent the blue and red components.

1.2.2 Chroma Subsampling

Because human eyes are more sensitive to the pixel intensities represented by the luminance change (Y) and less sensitive to the chrominance components in channel (C_b and C_r), it is common to further reduce the spatial dimension of chrominance channels by an optional factor denoted by $J : a : b$, where J is the width of the horizontal sampling reference (typically 4), a is the number of chrominance samples in the first row, and b is the number of changes in the chrominance samples between the first and second rows of J pixels. This operation further reduces the information to be encoded later in the following compression pipeline.

1.2.3 Block Splitting

Both the luminance channel and the two downsampled chrominance channels are further split into non-overlapping 8x8 macroblocks. The term *minimum coded unit* (MCU) refers to these blocks. The size of the MCU of the luminance channel is 8x8 and the size of the MCU of the chrominance channel is 16x16, if the chroma subsampling factor is set to 4:2:0.

1.2.4 Zero-centering

Since the 8-bit image has pixel intensities ranging from 0 to 255, we subtract 128 from these pixel values to make them center around 0 and reduce their dynamic range. We denote this operation as $\boldsymbol{S}$ and the resulting image as $\boldsymbol{x_s}$:

$$\boldsymbol{x_s} = \boldsymbol{S}(\boldsymbol{x_Y}) = \boldsymbol{x_Y} - 128 \tag{1.3}$$

The same operation is applied to chrominance channels.

1.2.5 Discrete Cosine Transform (DCT)

Following zero-centering, we apply 2D discrete cosine transform (2D DCT) to each non-overlapping 8x8 macroblock to obtain its 8x8 DCT coefficients. Discrete cosine transform is a technique used to transfer a signal from pixel domain to frequency domain, similar to discrete Fourier transform (DFT) but all frequency components are real values, as opposed to the complex numbers in DFT. The 1D DCT is commonly defined as

$$y(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) cos(\frac{\pi k(2n+1)}{2N}),\ 0 \leq k < N \tag{1.4}$$

where $x(n)$ is the input 1D signal with N samples and $y(k)$ is its frequency counterpart. The coefficient $\alpha(k)$ is defined as

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } k = 0, \\ \sqrt{\frac{2}{N}}, & \text{if } 1 \leq k < N \end{cases} \tag{1.5}$$

The inverse discrete cosine transform IDCT is used to transform the signal from frequency domain back to pixel domain, and is given by

$$x(n) = \sum_{k}^{N-1} \alpha(k) y(k) cos(\frac{\pi k(2n+1)}{2N}),\ 0 \leq n < N \tag{1.6}$$

Compared to DFT, DCT produces real numbers and has better energy compaction. Unlike DFT, DCT does not produce artificial high-frequency coefficients which was caused by the boundary effects, namely the abrupt transitions at the end of each interval.

Two-dimensional DCT (2D DCT) can be obtained by applying 1D DCT successively to the rows and columns of each block in the image, hence 2D

DCT is a separable transform. We exploit this property in Section 3.3.1 to speed up the calculation.

In JPEG compression 2D DCT is applied to each block in the image and we denote this operation as $\boldsymbol{DCT}$ and the resulting coefficient matrix as $\boldsymbol{x_d}$. This operation is formally defined as

$$\begin{aligned} \boldsymbol{x_d}(u,v) &= \boldsymbol{DCT}(\boldsymbol{x_s}) \\ &= \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7}\boldsymbol{x_s} cos[\frac{(2x+1)u\pi}{16}]cos[\frac{(2y+1)v\pi}{16}] \end{aligned} \tag{1.7}$$

where u and v are the horizontal and vertical spatial frequencies of the pixel inside each 8x8 block, i.e. $0 \leq u < 8$, $0 \leq v < 8$.

Parameter α here is a normalizing scale factor where

$$\alpha(u), \alpha(v) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } u = 0 \text{ or } v = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{1.8}$$

By multiplying this factor, the transformation becomes orthonormal. It is worth noting that this is a linear operation as $\boldsymbol{x_s}$ is not inside the cosine functions. This is imperative for residual learning to be possible as we describe in more detail in Chapter 3. After the $\boldsymbol{DCT}$ operation, we have transformed the image from pixel domain to DCT domain.

The top-left corner of the 8x8 DCT coefficients of each macroblock is regarded as the DC coefficients and usually has the largest magnitude, while the other 63 coefficients are referred to as the AC coefficients and have relatively smaller magnitude. The DC coefficients contain the low-frequency information, while the AC coefficients at the bottom-right corner have the most high-frequency information. The same transform is applied to both the

luminance and chrominance channels. A sample picture in both pixel and frequency domain is illustrated in Fig. 1.3.

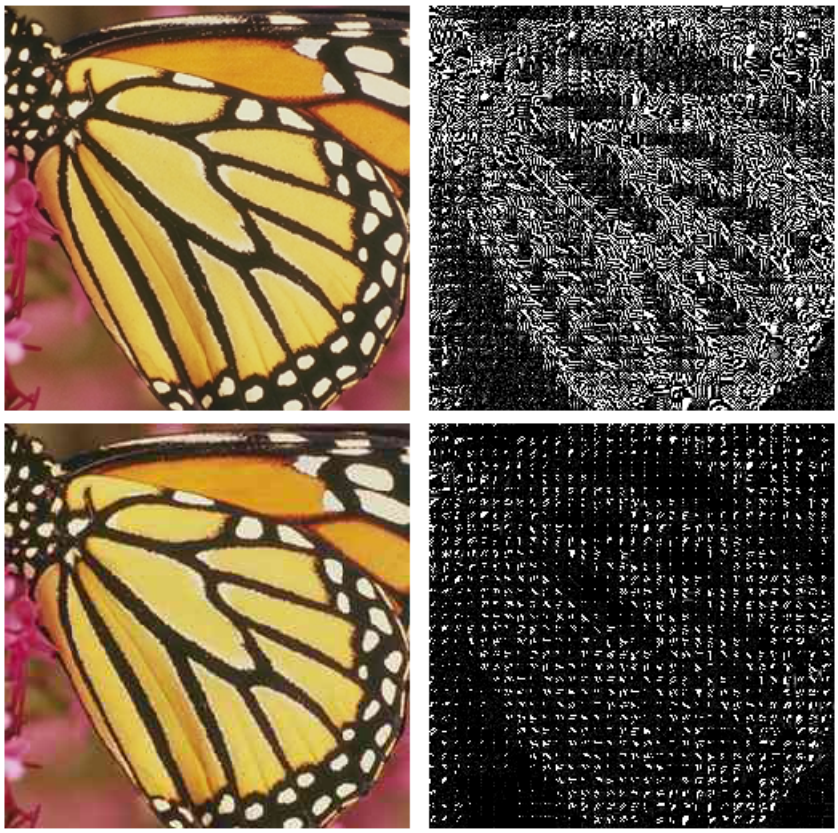

Figure 1.3: Top-left: Original image in pixel domain. Top-right: Original image in DCT domain. Bottom-left: Compressed image in pixel domain. Bottom-right: Compressed image in DCT domain.

1.2.6 Quantization

Considering that human eyes are not sensitive to brightness variations in a high-frequency area, we can leverage this insensitivity to reduce the information we need to encode by dropping certain high-frequency information in the bottom-right part of the 8x8 DCT blocks.. This is done by an element-wise division of the DCT coefficients in each 8x8 macroblock by a predefined

quantization table (Qt). We denote this operation as $\boldsymbol{Q}$ and the resulting quantized DCT coefficients as $\boldsymbol{x_q}$.

$$\boldsymbol{x_q}(i,j) = \boldsymbol{Q}(\boldsymbol{x_d}(i,j)) = \frac{\boldsymbol{x_d}(i,j)}{Qt(i,j)} \tag{1.9}$$

where the division is element-wise in each 8x8 DCT block and i, j denote the row and column indices respectively. This operation is applied to both luminance and chrominance channels by their corresponding quantization tables obtained using the method below.

Quantization tables are empirically found to yield satisfying quantization results and are sometimes trademarks for companies and entities who developed their own quantization tables. They also depend on the quality (QF) of the desired level of compression and differ in the luminance and chrominance channels. The commonly used quantization tables for QF of 50 are shown below.

$$Qt_{Y\_50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$Qt_{C\_50} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

After the element-wise division of the luminance and chrominance channels by their corresponding quantization tables, only a few coefficients in the top-left corner of each 8x8 DCT block, including the DC coefficient, remain large while all other AC coefficients become very close to 0.

Quantization tables for other quality factors QF_{new} can be obtained based on the two quantization tables for $QF = 50$ by the following equations:

$$S = \begin{cases} \frac{5000}{QF_{new}}, & \text{if } QF_{new} < 50 \\ 200 - 2 \cdot QF_{new}, & \text{if } QF_{new} > 50 \end{cases} \tag{1.10}$$

$$Qt_{Y\_new}(u, v) = \frac{S \cdot Qt_{Y\_50}(u, v) + 50}{100} \tag{1.11}$$

$$Qt_{C\_new}(u, v) = \frac{S \cdot Qt_{C\_50}(u, v) + 50}{100} \tag{1.12}$$

where u, v denote the row and column indices in each 8x8 DCT block.

1.2.7 Rounding

Following the previous step, we round all quantized DCT coefficients to the nearest integer, which results in most coefficients being 0. This operation

is denoted by $\boldsymbol{R}$ and resulting coefficients by $\boldsymbol{x_r}$.

$$\boldsymbol{x_r} = \boldsymbol{R}(\boldsymbol{x_q}) = round(\boldsymbol{x_q}) \tag{1.13}$$

This is the only operation in the JPEG compression pipeline that introduces loss, ignoring the numerical errors due to precision, which are negligible.

1.2.8 Entropy Coding

The last step in the encoding stage is entropy coding, which is a lossless operation applied on the rounded DCT coefficients. The DCT coefficients are first rearranged in a zigzag order to group similar frequencies together, illustrated in Fig. 1.4. This reordering captures non-zero coefficients first and adds zeros after the non-zero frequencies. Huffman coding is subsequently applied to the reordered coefficients to encode all non-zero frequencies.

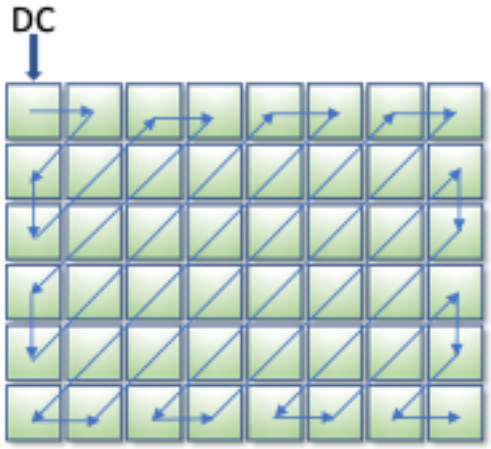

Figure 1.4: Illustration of zigzag ordering.

1.2.9 Decoding

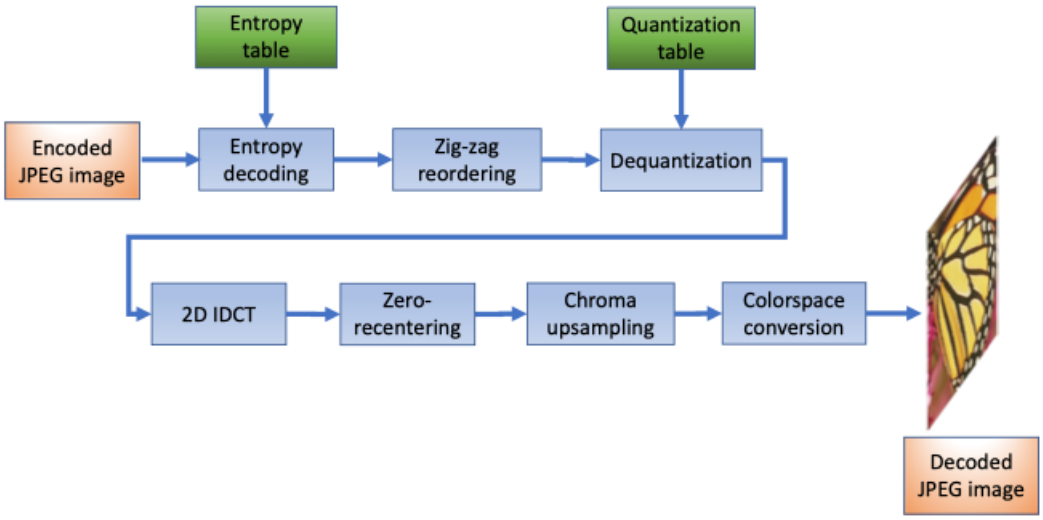

Figure 1.5: JPEG decoding pipeline.

The decoding process, shown in Fig. 1.5, applies all operations listed above in reverse order. After rearranging the DCT coefficient matrix in reverse zigzag order, we perform element-wise multiplication of the 8x8 macroblocks in the coefficient matrix with quantization tables of the corresponding channel. This operation is denoted by $\boldsymbol{Q^{-1}}$ and the resulting coefficient matrix $\tilde{\boldsymbol{x}}_{\boldsymbol{q}}$

$$\tilde{\boldsymbol{x}}_{\boldsymbol{q}}(i,j) = \boldsymbol{Q^{-1}}(\boldsymbol{x}_{\boldsymbol{r}}) = \boldsymbol{x}_{\boldsymbol{r}} * Qt(i,j) \tag{1.14}$$

where the multiplication is element-wise in each DCT block and i, j denote the row and column indices of each 8x8 block.

Due to the rounding operation $\boldsymbol{R}$ in the encoding stage, most elements in the coefficient matrix are zero at this point, which indicates the loss of high-frequency information is irreversible.

Two-dimensional inverse discrete cosine transform (2D IDCT) is then ap-

plied to each 8x8 block. This operation is denoted by $\boldsymbol{IDCT}$ and the resulting matrix in pixel domain by $\tilde{\boldsymbol{x}}_{\boldsymbol{d}}$.

$$\begin{aligned}\tilde{\boldsymbol{x}}_{\boldsymbol{d}}(i,j) &= \boldsymbol{IDCT}(\tilde{\boldsymbol{x}}_{\boldsymbol{q}}) \\ &= \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\tilde{\boldsymbol{x}}_{\boldsymbol{q}}\alpha(u)\alpha(v)cos[\frac{(2i+1)u\pi}{16}]cos[\frac{(2j+1)v\pi}{16}]\end{aligned} \tag{1.15}$$

where i and j are the horizontal and vertical indices of the pixel inside each 8x8 macroblock, i.e. $0 \leq i < 8$, $0 \leq j < 8$.

Parameter α is defined the same as Equation (1.5). Given that $\boldsymbol{IDCT}$ may result in non-integer values, we round the resulting coefficient matrix to the nearest integers. After this operation we have transformed from the DCT domain back to the pixel domain.

The last step in the decompression stage is to add 128 back to add pixel intensities. This operation is denoted by $\boldsymbol{S^{-1}}$ and the resulting image by $\hat{\boldsymbol{x}}$.

$$\hat{\boldsymbol{x}} = \boldsymbol{S^{-1}}(\tilde{\boldsymbol{x}}_{\boldsymbol{d}}) = \tilde{\boldsymbol{x}}_{\boldsymbol{d}} + 128 \tag{1.16}$$

Considering that this operation may result in pixel intensities outside of the range [0, 255], we clip $\hat{\boldsymbol{x}}$ by [0, 255] to make sure all pixel values are valid. At this point we have obtained the compressed image $\hat{\boldsymbol{x}}$ from the original image $\boldsymbol{x}$.

1.3 Compression Artifacts

As a lossy compression scheme, JPEG is able to produce compressed images with significantly fewer bits compared to the original images at the cost of visual degradation. The degradation is reflected by the artifacts coupled

with the compression process. Here we show a few compression artifacts that are commonly found in compressed images. The degree of corruption is dependent on the quality factor at the time of compression.

1.3.1 Blocking Artifacts

Due to the nature of JPEG compression that splits the image into 8x8 blocks, blocking artifacts arise as a result of inconsistency at the boundaries of macroblocks, due to the fact that adjacent pixels on either side of a boundary are being quantized into two different intensities. Blocking artifacts manifest as segmenting the original image into small square blocks, as shown in Fig. 1.6.

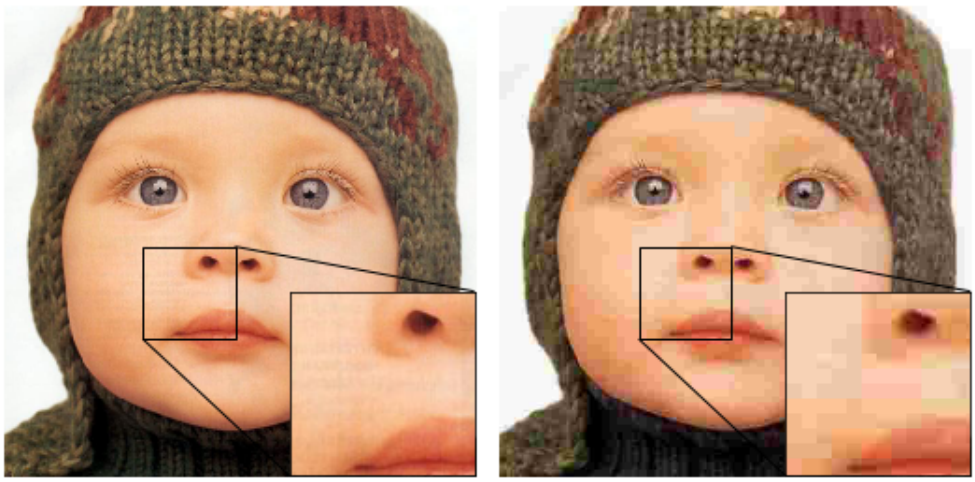

Figure 1.6: Left: Original image. Right: Compressed image with blocking artifacts. Best viewed in color.

1.3.2 Blurring Artifacts

As JPEG compression removes high-frequency components that are less noticeable to human eyes in the DCT domain, this loss is inevitably reflected in the compressed images, as can be seen in Fig. 1.7. The sharp edges, such as those on the woman's hat and hair, are smoothed out. It is worth mentioning that these discarded high-frequency components are found to be the most difficult to recover in existing and ongoing restoration approaches.

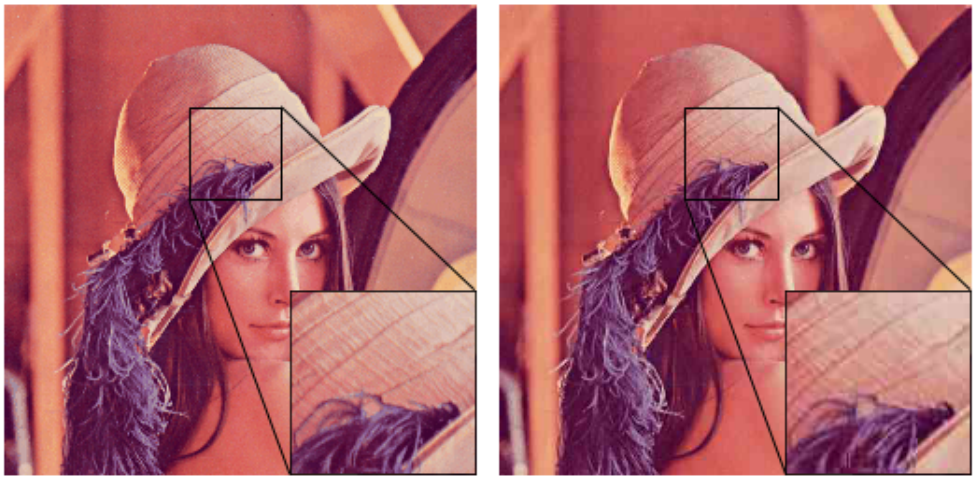

Figure 1.7: Left: Original image. Right: Compressed image with blurring artifacts. Best viewed in color.

1.3.3 Ringing Artifacts

For similar reasons, ringing artifacts arise around sharp edges due to the coarse quantization of high-frequency signal; see Fig. 1.8 These artifacts appear as oscillations around edges.

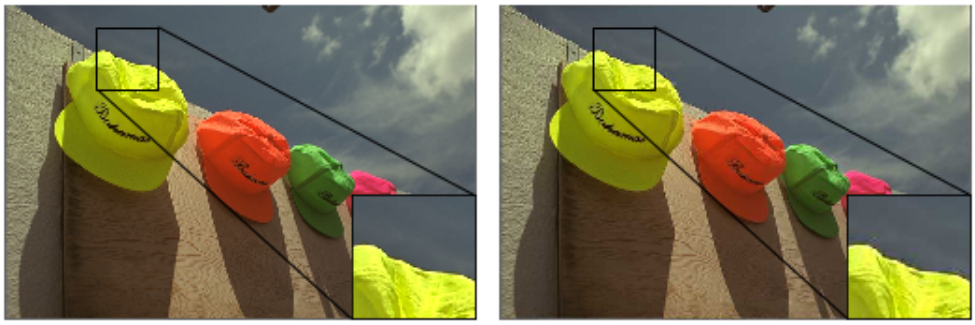

Figure 1.8: Left: Original image. Right: Compressed image with ringing artifacts. Best viewed in color.

1.3.4 Banding Artifacts

Coarse quantizations in smooth regions of an image may induce banding artifacts, or contouring artifacts, shown as the smooth regions being separated into visible bands; see Fig. 1.9 for illustration.

Figure 1.9: Left: Original image. Right: Compressed image with banding artifacts. Best viewed in color.

1.4 Convolutional Neural Network

Convolutional neural network (CNN) has enjoyed success in many visual imagery and natural language processing tasks over recent years due to its

exemplary representational power and ability to exploit spatial redundancies. It was first shown superior in image classification [1] and was later applied to various high-level computer vision tasks including object detection, recognition and segmentation. In recent years it has been shown that low-level computer vision tasks such as image super-resolution, denoising and reconstruction can also benefit from the power of CNNs.

CNN is a regularized version of multi-layer perceptrons which utilize convolutional neurons to convolve a filter with the signal in a predefined receptive field. They have a large amount of parameters, namely the weights and bias of the filters, that can be updated during gradient back-propagation to minimize the predefined loss between model output and the desired signal. Here we briefly review two CNN architectures that have demonstrated their superiority in various vision-related tasks, and are the inspiration behind the model we propose in this work.

1.4.1 ResNet

One way to better leverage the representational power of deep neural networks is to make them deeper, namely adding more convolution layers. Adding more layers effectively increases the amount of parameters to be learned and enables better approximation of the non-linear function the model is trying to represent. However, very deep neural networks are difficult to train as they do not converge easily. This is because the gradients calculated at back-propagation time are typically small and approach zero indefinitely when multiplied together; therefore, they cannot propagate back to the early layers. As a result, the neurons in early layers suffer from the vanishing gradient problem and are unable to update.

Deep residual neural network (ResNet) [2] was proposed to tackle the vanishing gradient problem by adding skip connections between early and later layers within each residual block. This is done by an element-wise addition of the input and the output of a subsequent convolution layer, as illustrated in Fig. 1.10.

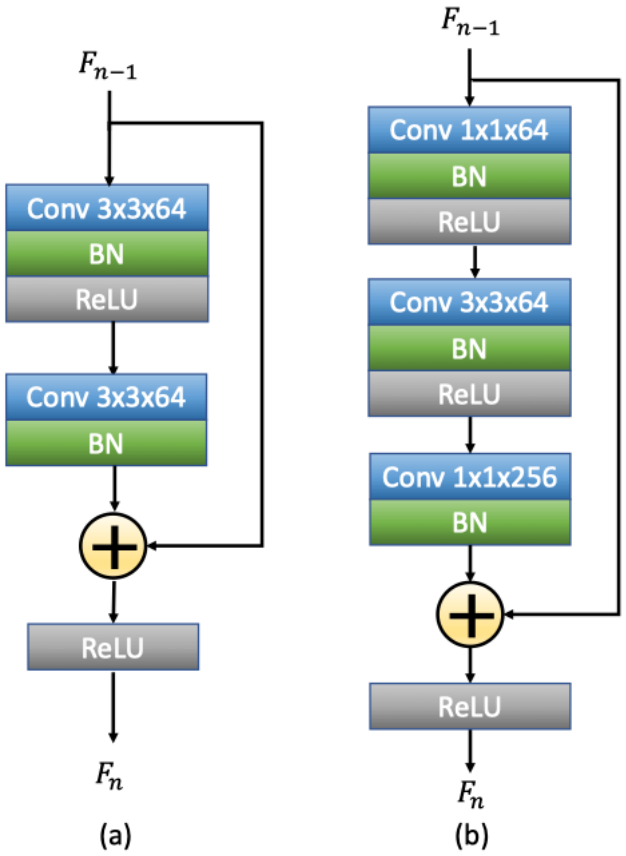

Figure 1.10: Two residual blocks proposed in [2].

The two proposed residual blocks add batch normalization after each convolution layer to reduce internal covariance shift and speed up training. The residual block in Fig. 1.10(b) replaces the first 3x3 convolution layer with a 1x1 layer and adds another 1x1 layer at the end of the residual block. Typically 1x1 convolution layers are used for feature dimensionality reduction, such that Fig. 1.10(b) reduces feature dimensionality first and then increases

it later within the same residual block.

1.4.2 Squeeze-and-excitation Network

Squeeze-and-excitation network [3] came under the spotlight for its significant improvement in ImageNet [4] classification accuracy. This network seeks to enhance spatial encoding to improve the representational power of the network. They exploit inter-channel dependencies and adaptively re-calibrate each channel of the feature maps. Specifically, the model learns to assign a weight to each channel based on its importance such that we can pay more attention to the more useful and informative channels and less attention to the other ones, as illustrated in Fig. 1.11.

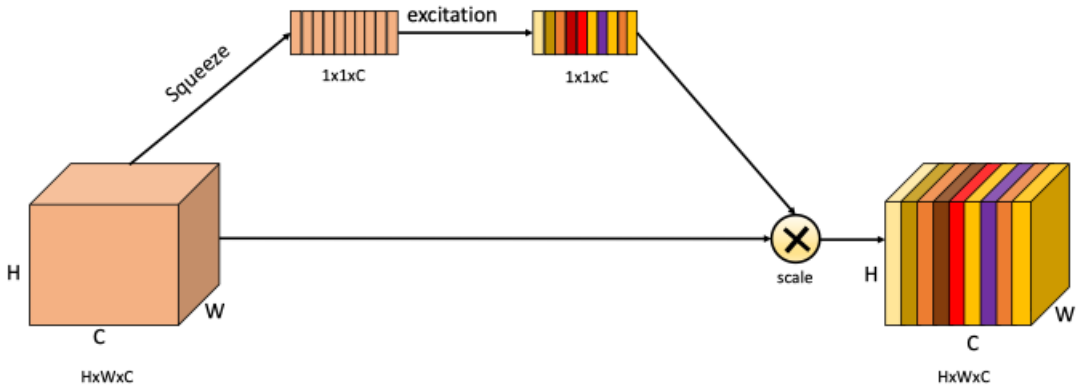

Figure 1.11: Illustration of squeeze-and-excitation module. Different color indicates different importance of each channel.

This module takes in a feature map from the previous layer and performs a channel-wise global average-pooling at the squeeze phase, which effectively reduces the dimension of the feature map from HxWxC to 1x1xC, followed by the excitation stage where the module transforms the 1x1xC vector to another 1x1xC vector which indicates the importance of each channel, as its importance factor. This importance factor is multiplied back to the in-

put of the squeeze-and-excitation module by each channel to produce the recalibrated feature map. Detailed operation of this module is shown in Fig. 1.12.

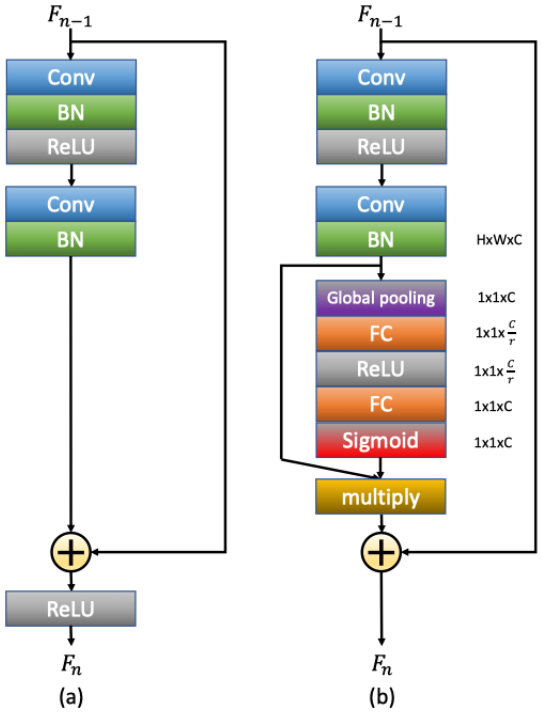

Figure 1.12: Left: Standard residual block. Right: Squeeze-and-excitation block.

Compared to the standard residual block in ResNet (Fig. 1.12(a)), the squeeze-and-excitation block does not add too much computation but is better able to capture global information outside of the receptive field than simply stacking convolutional filters. By embedding the global information in the feature maps, various classification-related tasks are better able to reduce their classification errors.

1.5 Performance Measurement

After enhancing or reconstructing images, it is common to adopt objective evaluation metrics to quantify the enhancement or improvement. Besides

mean squared error, which indicates the squared difference between an image and its reference, popular evaluation metrics include peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [5] and PSNR-B [6], which is specifically designed to assess blocking artifacts in JPEG compressed images by including a blocking effect factor (BEF). These three commonly used metrics are described in detail in the following sections.

1.5.1 PSNR

The mean square error (MSE) and peak signal-to-noise ratio (PSNR) are two commonly used error metrics. MSE measures the cumulative squared error between an image I and its reference image R while PSNR gives the peak error. MSE is formally defined as:

$$MSE(I,R) = \frac{\sum_{M,N}(I(m,n) - R(m,n)^2}{M \times N} \tag{1.17}$$

and PSNR is given by

$$PSNR(I,R) = 10 \times log_{10}(\frac{A^2}{MSE(I,R)}) \tag{1.18}$$

correspondingly, where A is the maximum possible pixel intensity value, typically 1 for images where all pixel intensities are between 0 and 1, or 255 for images where all pixel intensities are between 0 and 255.

Typically images with better quality have a lower MSE and higher PSNR, and two identical images would have an MSE of 0 and PSNR of infinity. In addition, because human eyes are more sensitive to luminance components and less sensitive to chrominance components, it is sometimes common to evaluate PSNR on luminance channel only instead of the RGB image.

1.5.2 SSIM

Although commonly adopted for image quality measurement, MSE and PSNR sometimes fail to correlate with the perceptible quality loss and are therefore not entirely reliable. Structural similarity index (SSIM) [5], on the other hand, assesses image degradation by measuring the perceptible loss by considering structural information and perceptual phenomena. It exploits the structure manifested as inter-dependencies among spatially close pixels. It also considers luminance masking and contrast masking, the former indicating a phenomenon where image distortions are more conspicuous in bright areas and the latter indicating distortions are also less visible in areas with more textures.

Mathematically SSIM is a product of three different aspects of similarity: luminance, contrast and structure. The luminance comparison of image I and reference image R is defined as:

$$l(I,R) = \frac{2\mu_I\mu_R + C_1}{\mu_I^2 + \mu_R^2 + C_1} \tag{1.19}$$

where μ_x and μ_y are mean of I and R, and C_1 is a stabilizing constant.

Similarly the contrast comparison is defined as:

$$c(I,R) = \frac{2\sigma_I\sigma_R + C_2}{\sigma_I^2 + \sigma_R^2 + C_2} \tag{1.20}$$

where σ_I and σ_R are standard deviation of I and R, and C_2 is a stabilizing constant.

The structure comparison is defined as:

$$s(I,R) = \frac{\sigma_{IR} + C_3}{\sigma_{IR} + C_3} \tag{1.21}$$

where σ_{IR} is the correlation between I and R, and C_3 is another stabilizing constant.

Putting it together, the SSIM is given by:

$$SSIM(I,R) = l(I,R)^{\alpha} \times c(I,R)^{\beta} \times s(I,R)^{\gamma} \tag{1.22}$$

In most cases we have $C_3 = \frac{C_2}{2}$ and $\alpha = \beta = \gamma = 1$, and the SSIM equation is reduced to a common form:

$$SSIM(I,R) = \frac{(2\mu_I\mu_R + C_1)(2\sigma_{IR} + C_2)}{(\mu_I^2 + \mu_R^2 + C_1)(\sigma_I^2 + \sigma_R^2 + C_2)} \tag{1.23}$$

1.5.3 PSNR-B

To measure the image degradation caused by JPEG compression, specifically the severity of blocking artifacts, PSNR-B [6] was designed to incorporate a blocky effect factor term that measures the differences in pixel intensities across DCT blocks, specifically the change in luminance levels around block boundaries. This metric simply measures the extent of apparent blocking artifacts and does not require a reference image.

Assume we have an image I with shape $N_V \times N_H$. Let H and V be the sets of all pairs of adjacent pixels in image I in horizontal and vertical directions. Denote H_B to be a set of all pairs of adjacent pixels across the horizontal boundaries and V_B to be a set of all pairs of pixels across the vertical boundaries. Consequently denote the complements of H_B and V_B, which are all pairs of adjacent pixels that do not lie on the horizontal or vertical boundaries, as H_B^C and V_B^C, i.e.

$$H_B^C = H - H_B V_B^C = H - V_B \tag{1.24}$$

I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}
I_{17}	I_{18}	I_{19}	I_{20}	I_{21}	I_{22}	I_{23}	I_{24}
I_{25}	I_{26}	I_{27}	I_{28}	I_{29}	I_{30}	I_{31}	I_{32}
I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}	I_{40}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}
I_{49}	I_{50}	I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}
I_{57}	I_{58}	I_{59}	I_{60}	I_{61}	I_{62}	I_{63}	I_{64}

Figure 1.13: Illustration of pixel blocks.

An illustration is shown in Fig. 1.13, where

$$\begin{aligned} H_B &= \{(I_4, I_5), (I_{12}, I_{13}), ...\} \\ H_B^C &= \{(I_1, I_2), (I_2, I_3), ...\} \\ V_B &= \{(I_25, I_33), (I_{26}, I_{34}), ...\} \\ V_B^C &= \{(I_1, I_9), (I_9, I_{17}), ...\} \end{aligned} \tag{1.25}$$

Denoting the number of pairs in sets H_B, H_B^C, V_B, V_B^C by N_{H_B}, $N_{H_B}^C$, N_{V_B}, $N_{V_B}^C$, respectively, and the block size by B, then

$$\begin{aligned} N_{H_B} &= N_V[\frac{N_H}{B} - 1] \\ N_{H_B}^C &= N_V(N_H - 1) - N_{H_B} \\ N_{V_B} &= N_H[\frac{N_V}{B} - 1] \\ N_{V_B}^C &= N_H(N_V - 1) - N_{V_B} \end{aligned} \tag{1.26}$$

We further differentiate the squared difference among all pixel pairs lying

on the boundaries and pixel pairs not lying on the boundaries by:

$$D_B(I) = \frac{\sum_{(I_u,I_v)\in H_B}(I_u - I_v)^2 + \sum_{(I_u,I_v)\in V_B}(I_u - I_v)^2}{N_{H_B} + N_{V_B}} \tag{1.27}$$

$$D_B^C(I) = \frac{\sum_{(I_u,I_v)\in H_B^C}(I_u - I_v)^2 + \sum_{(I_u,I_v)\in V_B^C}(I_u - I_v)^2}{N_{H_B^C} + N_{V_B^C}} \tag{1.28}$$

where D_B and D_B^C are mean boundary pixel squared difference and mean nonboundary pixel squared difference, respectively. As the quantization step increases, D_B will approach D_B^C, in which case the blocking artifacts will become more obvious.

We also define blocking effect factor (BEF) as

$$BEF(I) = \eta \times [D_B(I) - D_B^C(I)] \tag{1.29}$$

where

$$\eta = \begin{cases} \frac{log_2^B}{log_2(min(N_H,N_V))}, & \text{if } D_B(I) > D_B^C(I) \\ 0, & \text{otherwise} \end{cases} \tag{1.30}$$

from which we can see that η is a function of block size B, and the resulting PSNR-B is nonzero if and only if the differences across pixel pairs lying on block boundaries are larger than the differences across non-boundary pixel pairs.

Finally the PSNR-B is defined as:

$$MSE - B(I,R) = MSE(I,R) + BEF(I) \tag{1.31}$$

$$PSNR - B(I,R) = 10log_{10}\frac{255^2}{MSE - B(I,R)} \tag{1.32}$$

CHAPTER 2

RELATED WORK

Existing works targeting suppression of compression artifacts have been developed using various approaches.

Conventional approaches treat artifact removal as either a deblocking-oriented or restoration-oriented task [7]. Deblocking-oriented methods aim at removing blocking artifacts in spatial domain [8, 9] by devising adaptive filters to remove blocking artifacts in specific regions, or they denoise by thresholding in the wavelet domain [10]. These approaches tend to oversmooth and fail to recover high-frequency information such as edges in smooth regions due to filtering operations. Restoration-oriented methods, including projection onto convex sets (POCS) [11] and regression tree fields [12], aim to directly remove distortion introduced at compression time by utilizing prior knowledge.

Learning based approaches, on the other hand, have demonstrated superior reconstruction quality. Two representative learning based approaches include sparse-coding based methods and reconstruction with a CNN. Sparsity-based works seek to reconstruct the original image by building a dictionary of the sparse representation of a set of compressed images using K-singular vector decomposition (K-SVD) algorithm, then estimating the original image from the learned dictionary by constraining the errors [13]. Liu et al. [14] incorporated prior knowledge of JPEG compression by carrying out the sparse-coding in both pixel and DCT domain to constrain the quantization errors

within DCT domain. Both works have shown impressive deblocking performance, but the performance is limited by the number and size of dictionaries, and is found to be accompanied by either noisy edges or over-smoothed regions [15]. Another drawback of these sparsity-based approaches is that the reconstruction stage is performed iteratively, which is computationally expensive.

Since their success at the AlexNet [1] in ImageNet classification tasks [4], convolutional neural networks (CNNs) have enjoyed continuous success at various high-level computer vision tasks including classification, segmentation and recognition, etc. In recent years CNNs have also been shown to succeed in low-level computer vision tasks such as image restoration and enhancement. Dong et al. first demonstrated that CNNs can be utilized to restore images [16]. They approached the task of image super-resolution by formulating a three-layer CNN (SRCNN) which is tailored to perform the same jobs that sparsity-based models do. Specifically, the three convolutional layers aim to extract and represent features, perform non-linear mapping and reconstruct images, respectively. Although this three-layer network is effective at recovering high-resolution details, the limit in depth prevents its further improvement. Kim et al. [17] developed a significantly deeper CNN (VDSR) with 20 convolutional layers to perform single-image super-resolution task. They ensured that by only learning the residual of the input images, gradients can be back-propagated to the early layers and not lost in the deep network. Based on SRCNN, Dong et al. [18] showed that compression artifacts can be eliminated by enhancing the features extracted by a CNN, and proposed a network (ARCNN) specifically targeting removal of compression artifacts. ARCNN adds an extra convolutional layer after the feature extraction layer to enhance extracted features. They also demon-

strated that directly training a model to recover images compressed with a lower QF is difficult; however, by applying transfer learning, specifically fine-tuning a model pretrained on recovering images with a relatively higher QF, their model is able to perform better on more difficult tasks. Inspired by the dual-domain sparsity-coding approach, Guo and Chao [15] developed a CNN (DDCN) while applying the constraint on quantization loss in DCT domain to better reconstruct original images. DDCN learns residual images in both DCT and pixel domain, and aggregates outputs from these two domains with additional convolutional layers to produce the final output. Their work also demonstrated the importance of the Adam optimization method [19], which restricts gradient update in each backpropagation step in order to avoid exploding gradient and ease the training of a very deep network. It is worth noting that all these CNN based approaches focus on recovering a grayscale image or the luminance channel of an image; however, this is not the solution to the more practical problem, which is to reconstruct high-quality color images to be displayed by users in high-resolution displays.

CHAPTER 3

APPROACH

Conventional learning-based approaches have been focusing on recovering a grayscale image or the luminance channel of a color image; however, recovering color images poses additional challenges as compression artifacts manifest differently in luminance and chrominance channels, as shown in the examples in Fig. 3.1 and Fig. 3.2.

Figure 3.1: Comparison of compression artifacts in different channels.

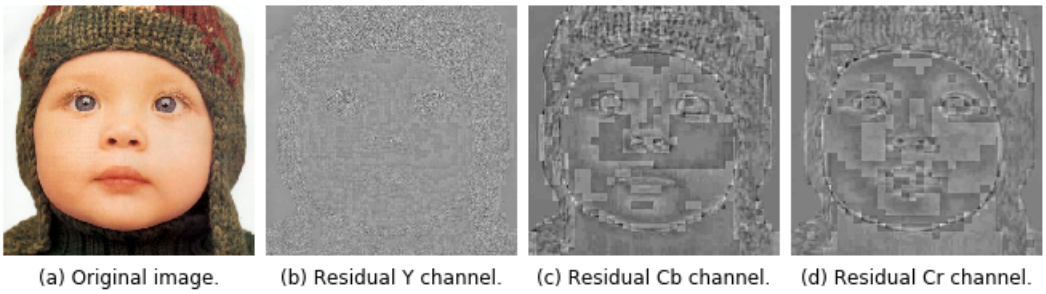

Figure 3.2: Comparison of compression artifacts in residual images.

In order to suppress JPEG compression artifacts and recover high-quality color images, we first adapt prior work that focuses on simply recovering

a grayscale image or the luminance channel of an image, then take advantage of the prior knowledge of JPEG compression by developing a dual-domain CNN to fully confine the quantization loss within DCT domain and recover color images. We propose a residual block tailored for artifact removal by utilizing the representational power of ResNet [2] to tackle the vanishing gradient problem and better explore redundancies and use squeeze-and-excitation technique to further extract useful information across different channels from the feature map. We call this residual block dual-domain squeeze-and-excitation artifact removal ResNet (SE-ARResNet), and incorporate this block into our proposed network.

3.1 Super-Resolution Networks

We first adapt CNN used for single-image super-resolution task to remove compression artifacts of color images. We start with the SRCNN architecture described in [16] and fix the input channel of the first feature extraction layer and the output channel of the last reconstruction layer to be three. ReLU activation layer is applied after the feature extraction and non-linearity mapping layer, and no activation is applied at the last reconstruction layer. The network architecture is shown in Fig. 3.3. For each convolutional layer, the first two numbers denote filter height and width and the last two numbers denote the number of input and output channels. As with the SRCNN described in the previous section, the number of input channels of the first convolutional layer and the number of output channels of the last convolutional layer are adjusted to 3 to accommodate the number of channels of color images.

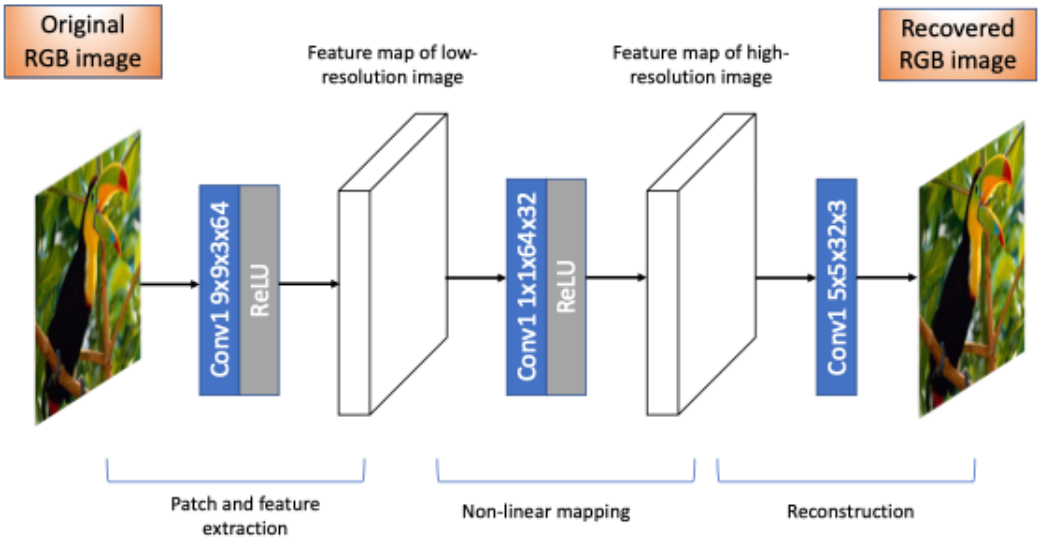

Figure 3.3: Network architecture of SRCNN.

The network is optimized by minimizing the l2-loss between the output image from the network $\hat{\boldsymbol{y}}^{(i)} = F(\hat{\boldsymbol{x}}^{(i)}; \Theta)$ and the ground-truth original image $\boldsymbol{x}^{(i)}$:

$$Loss(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \left\| \hat{\boldsymbol{y}}^{(i)} - \boldsymbol{x}^{(i)} \right\|_2^2 \tag{3.1}$$

where m is the total number of training images, and Θ denotes the trainable parameters in this network including filter weights and biases.

Given that SRCNN has only three convolutional layers, it may not have enough representational power to tackle the compression artifact suppression task; therefore, we conduct a second experiment with a super-resolution network, VDSR [17], which has 20 convolutional layers. Instead of learning to recover the entire original image, VDSR learns the residual image, or the correction image. We denote the ground-truth residual image as $\boldsymbol{r}$ and the learned residual image as $\hat{\boldsymbol{r}}$:

$$\boldsymbol{r} = \boldsymbol{x} - \hat{\boldsymbol{x}} \tag{3.2}$$

A sample residual image is illustrated in Fig. 3.4. Ideally if the network can fully recover the residual image, we can reconstruct the original uncompressed image completely.

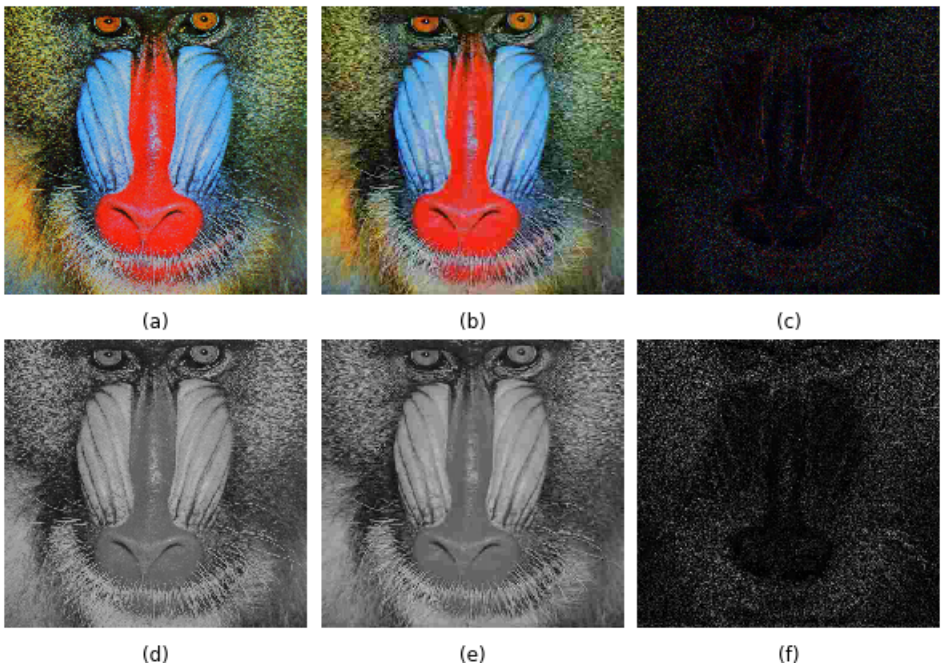

Figure 3.4: Residual image in RGB and luminance channels. (a) Original RGB image. (b) Compressed RGB image. (c) RGB residual image. (d) Luminance channel of the original image. (e) Luminance channel of the compressed image. (f) Luminance channel of the residual image.

As the last operation of the network, the residual image is added back to the input image to produce the output, as illustrated in Fig. 3.5.

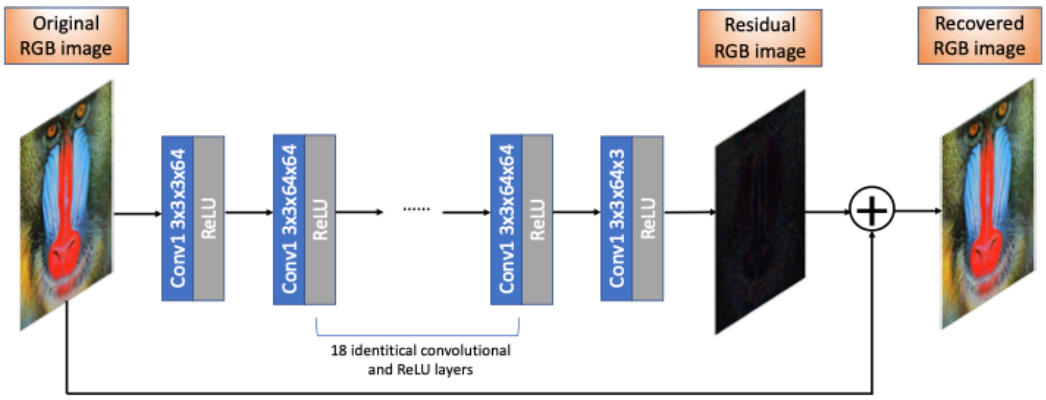

Figure 3.5: Network architecture of VDSR.

The only difference in this network from its original design [17] is that the number of input channels of the first convolutional layer and the number of output channels of the last convolutional layer are both 3, corresponding to the number of channels of color images.

Mathematically learning the residual image can be represented by modifying the loss function to minimize the l2-loss between network output and the ground-truth residual image $\boldsymbol{r}^{(i)}$:

$$\begin{aligned} Loss(\Theta) &= \frac{1}{m}\sum_{i=1}^{m}\left\|\hat{\boldsymbol{y}}^{(i)} - (\boldsymbol{x}^{(i)} - \hat{\boldsymbol{x}}^{(i)})\right\|_2^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}\left\|\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{r}^{(i)}\right\|_2^2 \end{aligned} \tag{3.3}$$

3.2 Artifact Removal Networks

As Dong et al. [18] pointed out, compression artifacts degrade the original image in pixel domain and introduce noise to the features extracted the by convolutional operation. To remove the added noise, the artifact removal network (ARCNN) has an extra feature enhancement convolutional layer following the feature extraction layer, right before the non-linear mapping

layer.

We experimented with ARCNN to reconstruct color images by changing the input and output channels to 3 to recover color images. The ARCNN architecture is illustrated in Fig. 3.6.

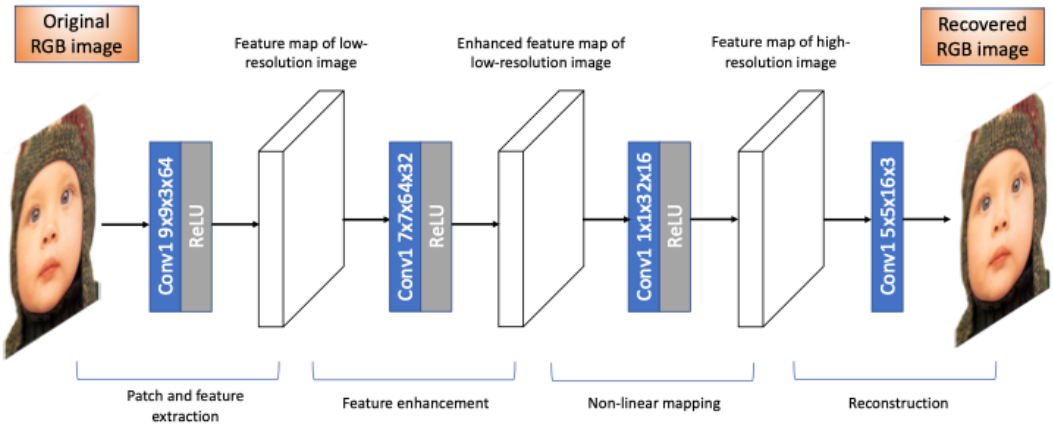

Figure 3.6: Network architecture of ARCNN.

Since this network learns to reconstruct the full image directly, we minimize the same loss function as in SRCNN.

3.3 Dual-domain Approach

In order to incorporate prior knowledge of JPEG compression in frequency domain to recover the original image in pixel domain, we follow the dual-domain approach proposed in DDCN [15] and design a new model in which we train separate CNNs in pixel and frequency domains and fuse the output from the two domains to produce the final output with the help of ResNet [2] and squeeze-and-excitation network [3].

3.3.1 DCT Branch

As noted in Chapter 1, the only operation in JPEG compression is the rounding operation, which brings in quantization errors for the sake of efficient encoding. Specifically, we devise a DCT branch by using a CNN in DCT domain (denoted by $CONV_{D\{i\}}$) to learn DCT coefficients and apply the constraint on quantization error to make sure the errors are not propagated to the output. Because DCT coefficients are ordered corresponding to their frequencies, exploiting the redundancies in frequency domain is beneficial to recovering high-frequency information. Theoretically, if the CNN in DCT branch is able to recover the DCT coefficients of the original image, we can recover the original image with very high reconstruction quality. Therefore, the goal of the DCT branch is to leverage the representational power of the CNN to learn and minimize the MSE between the quantized DCT coefficients of ground-truth $\boldsymbol{x}$ and the quantized DCT coefficients of output image $\hat{\boldsymbol{y}}$. Since all operations in JPEG compression besides rounding are invertible, we are able to obtain the quantized DCT coefficients of the DCT domain output images by applying the compression procedure in reverse order. Specifically,

$$\begin{aligned} \hat{\boldsymbol{y}}_{\boldsymbol{d}} &= \boldsymbol{DCT}(\boldsymbol{S}(\hat{\boldsymbol{y}})) \\ &= \boldsymbol{DCT}(\hat{\boldsymbol{y}} - 128) \end{aligned} \tag{3.4}$$

where $\hat{\boldsymbol{y}}$ is the output of DCT domain in YCbCr colorspace and we assume these operations are applied to both luminance and chrominance channels.

Similarly we can obtain the quantized DCT coefficients of the original image by:

$$\begin{aligned} \boldsymbol{x_d} &= \boldsymbol{DCT}(\boldsymbol{S}(\boldsymbol{x})) \\ &= \boldsymbol{DCT}(\boldsymbol{x} - 128) \end{aligned} \tag{3.5}$$

Since all operations are linear in Equation (3.5) above, minimizing the MSE of quantized DCT coefficients $\hat{\boldsymbol{y}}_{\boldsymbol{q}}$ and $\boldsymbol{x_q}$ is the same as minimizing the MSE between $\hat{\boldsymbol{y}}$ and $\boldsymbol{x}$.

That said, we build the DCT branch by putting a DCT layer and an IDCT layer at the beginning and end of the DCT branch, with a CNN between the two. In practice, DCT and IDCT layers convolve feature maps with a predefined 2D DCT coefficient kernel $\boldsymbol{W_{dct\_kernel}}$ to fully utilize the computational speedup enabled by GPUs. We reshape the regular 1D DCT coefficient matrix of size 64 to a 2D matrix of size 8×8, and denote it as W_{dct}. Then the 2D DCT coefficient matrix can be obtained by taking the Kronecker product of W_{dct}:

$$\boldsymbol{W_{dct\_kernel}} = W_{dct} \otimes W_{dct} \tag{3.6}$$

which results in a 64×64 kernel.

3.3.2 Pixel Branch

Parallel to the DCT branch, we use another CNN (denoted by $CONV_{P\{i\}}$) in the pixel branch to directly exploit spatial redundancies, as learning in pixel domain has been shown necessary and efficient for removing blocking artifacts in our early experiments. This is because while transforming the signal from pixel domain to frequency domain via DCT, the spatial information is lost. The pixel branch addresses this problem and complements the

DCT branch to exploit both spatial and frequency redundancies. We feed the RGB compressed image to this branch for it to learn the RGB residual image.

3.3.3 Aggregations Branch

The output of DCT and pixel branches are concatenated before feeding into another aggregation network, which is also a CNN (denoted by $CONV_{A\{i\}}$) and outputs the final residual image. The input compressed RGB image is then added to the output residual image by element-wise addition to produce the final recovered image.

3.3.4 Feature Representation in DCT Branch

In our previous experiment on removing compression artifacts with CNNs, we noticed that blocking and ringing artifacts are relatively easy to eliminate but high-frequency information is hard to recover. As a result we propose to utilize the CNN in the DCT branch to better capture correlations with high-frequency information. This is achieved by reshaping each 8x8 macroblock of one channel in the DCT input to a vector with shape 1x1x64 with the patch layer placed at the beginning of the DCT branch. The patch layer convolves a predefined one-hot kernel to the input image $\hat{\boldsymbol{x}}$ and outputs a spatially downsampled (by a factor of 8) tensor with 64 channels. For example, the reshaped DCT coefficient matrix of image of size 512x512 will become 64x64x64. In this case each 1x1x64 vector represents an 8x8 macroblock in the original image while the first channel represents the DC components of each macroblock, which captures the low-frequency components of the image, while the last channel captures only high-frequency information of the

original image. A sample image is shown in Fig. 3.7 with the first and last three channels of the DCT feature map.

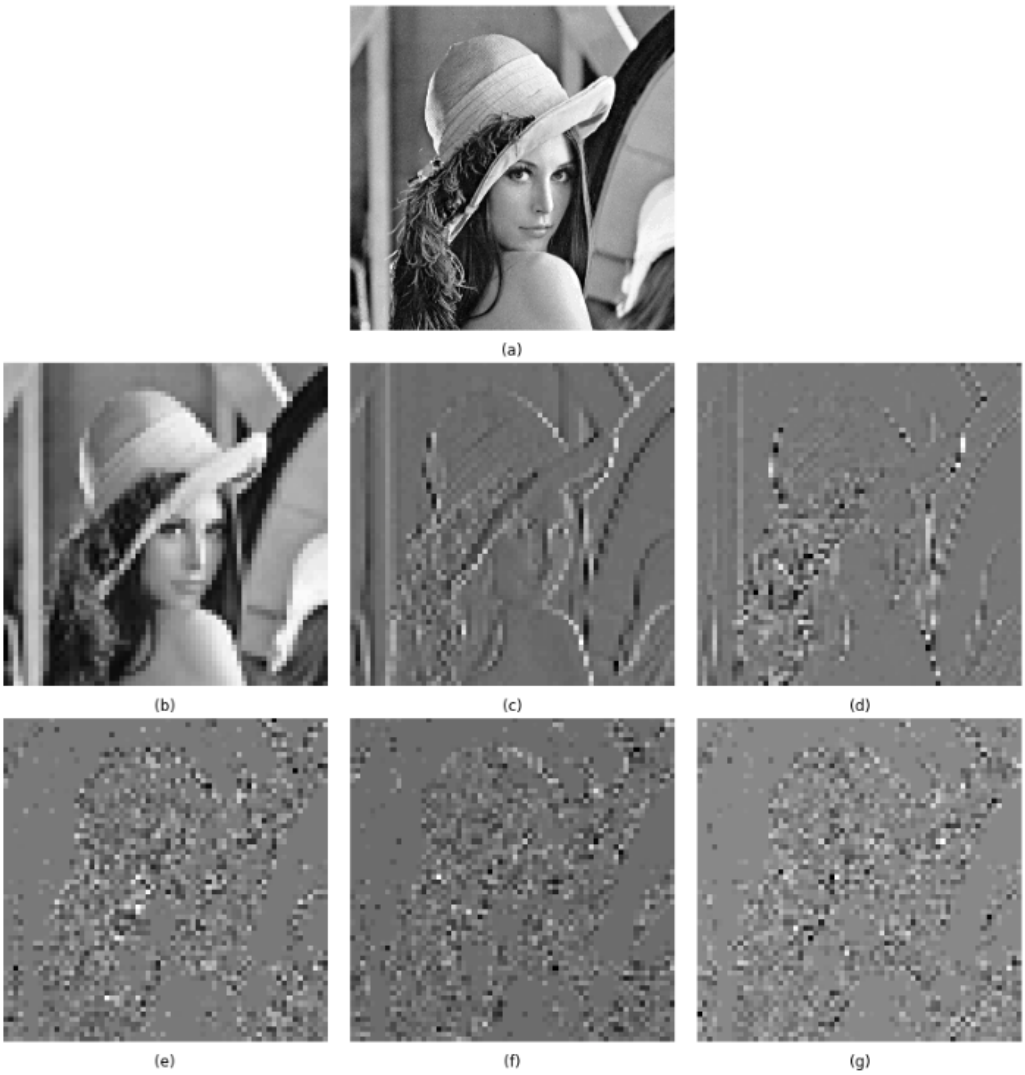

Figure 3.7: Sample DCT coefficients of different channels. (a) DCT coefficients of the luminance channel (512x512). (b) 1st channel of the DCT coefficients (64x64). (c) 2nd channel of the DCT coefficients (64x64). (d) 3rd channel of CDT coefficients (64x64). (e) 62nd channel of DCT coefficients (64x64). (f) 63rd channel of DCT coefficients (64x64). (g) 64th channel of DCT coefficients (64x64).

The DDCN [15] claims that image patches should not be extracted aligning with the 8x8 macroblocks boundaries, but randomly extracted with the possibility of misaligning with macroblocks, in that it helps with remov-

ing the blocking artifacts. We found empirically that although this misaligned extraction may better suppress blocking artifacts, it misrepresents high-frequency information in the DCT domain, which is a much bigger problem than removing blocking artifacts. This misrepresentation is illustrated in Fig. 3.8.

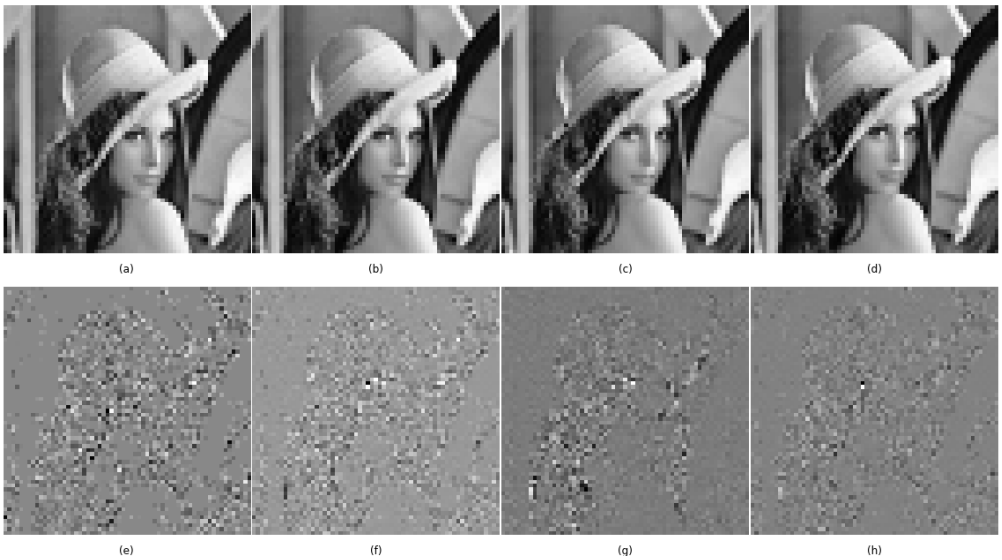

Figure 3.8: Comparison of DCT Coefficients At Different Alignment. (a) 1st channel of DCT coefficients aligned with DCT blocks. (b) 1st channel of DCT coefficients misaligned with DCT blocks by 1 pixel. (c) 1st channel of DCT coefficients misaligned with DCT blocks by 4 pixels. (d) 1st channel of DCT coefficients misaligned with DCT blocks by 7 pixels. (e) Last channel of DCT coefficients aligned with DCT blocks. (f) Last channel of DCT coefficients misaligned with DCT blocks by 1 pixel. (g) Last channel of DCT coefficients misaligned with DCT blocks by 4 pixels. (h) Last channel of DCT coefficients misaligned with DCT blocks by 7 pixels.

In addition, an unpatch layer is placed at the very end of the DCT branch to transform the frequency features back into the original dimensions of the input image. The 64x64x64 feature map in frequency domain mentioned earlier will be transformed back to 512x512x1.

3.3.5 Quantization Error Constraint

Given that the quantized DCT coefficients are being rounded to the nearest integer, it easy to see that the difference between coefficients before and after the rounding operation should be always below 0.5:

$$\left|\boldsymbol{x_q} - \boldsymbol{x_r}\right| \leq \frac{1}{2} \tag{3.7}$$

meaning that given $\boldsymbol{x_r}$, we can determine the valid range of the DCT coefficients of the original image. Therefore we can leverage this prior knowledge by constraining the CNN output in the DCT branch to be within the following range:

$$\hat{\boldsymbol{y}}_{\boldsymbol{q}} - \frac{1}{2} \leq \boldsymbol{x_q} \leq \hat{\boldsymbol{y}}_{\boldsymbol{q}} + \frac{1}{2} \tag{3.8}$$

3.3.6 Residual Learning

Following the success of VDSR at rapid convergence, we apply residual learning in this network as well. This is theoretically possible because we can rewrite Equation (3.8) as:

$$\begin{gathered} -\frac{1}{2} \leq \boldsymbol{x_q} - \hat{\boldsymbol{y}}_{\boldsymbol{q}} \leq \frac{1}{2} \\ -\frac{1}{2} \leq \boldsymbol{Q}(\boldsymbol{DCT}(\boldsymbol{S}(\boldsymbol{x}))) - \boldsymbol{Q}(\boldsymbol{DCT}(\boldsymbol{S}(\hat{\boldsymbol{y}}))) \leq \frac{1}{2} \\ -\frac{1}{2} \leq \frac{\boldsymbol{DCT}(\boldsymbol{x} - 128)}{Qt} - \frac{\boldsymbol{DCT}(\hat{\boldsymbol{y}} - 128)}{Qt} \leq \frac{1}{2} \end{gathered} \tag{3.9}$$

Since all operations in Equation (3.9) are linear, we can further simplify it to:

$$-\frac{1}{2} \leq \frac{\boldsymbol{DCT}(\boldsymbol{x} - \hat{\boldsymbol{y}})}{Qt} \leq \frac{1}{2} \tag{3.10}$$

where $\boldsymbol{x} - \hat{\boldsymbol{y}}$ is the residual image.

In practice since all input and ground-truth images are normalized by 255 to be within the range [0, 1], we need to normalize the quantization tables by 255 as well such that Equation (3.10) still holds.

The error constraint in Equation (3.10) can be easily achieved by a clipping layer after the last convolutional layer in the DCT branch, and we denote the output of the last convolutional layer as $\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}$ and output of the clipping layer as $\tilde{\boldsymbol{x}}_{\boldsymbol{clip}}$:

$$\tilde{\boldsymbol{x}}_{\boldsymbol{clip}} = \begin{cases} -\frac{1}{2}, & \text{if } \tilde{\boldsymbol{x}}_{\boldsymbol{conv}} < -\frac{1}{2} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, & \text{if } -\frac{1}{2} \leq \tilde{\boldsymbol{x}}_{\boldsymbol{conv}} \leq \frac{1}{2} \\ \frac{1}{2}, & \text{if } \tilde{\boldsymbol{x}}_{\boldsymbol{conv}} > \frac{1}{2} \end{cases} \tag{3.11}$$

This clipping operation is linear between -0.5 and 0.5 but flat outside this range. Since the flat regions have a derivative of 0 and may lead to vanishing gradient problems, we add a small fixed slope (α) to the flat regions such that their gradients are not 0 and call this layer the adaptive clipping layer. Therefore Equation (3.11) is updated to:

$$\tilde{\boldsymbol{x}}_{\boldsymbol{clip}} = \begin{cases} (1-\alpha) * (-\frac{1}{2}) + \alpha * \tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, & \text{if } \tilde{\boldsymbol{x}}_{\boldsymbol{conv}} < -\frac{1}{2} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, & \text{if } -\frac{1}{2} \leq \tilde{\boldsymbol{x}}_{\boldsymbol{conv}} \leq \frac{1}{2} \\ (1-\alpha) * \frac{1}{2} + \alpha * \tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, & \text{if } \tilde{\boldsymbol{x}}_{\boldsymbol{conv}} > \frac{1}{2} \end{cases} \tag{3.12}$$

This operation is illustrated in Fig. 3.9. Here we initialize α to be 0.1 and make it learnable, meaning its value will be updated during back-propagation. Additionally, instead of having one unique α in the entire model, we train this variable for every channel of the output from the last DCT convolutional layer.

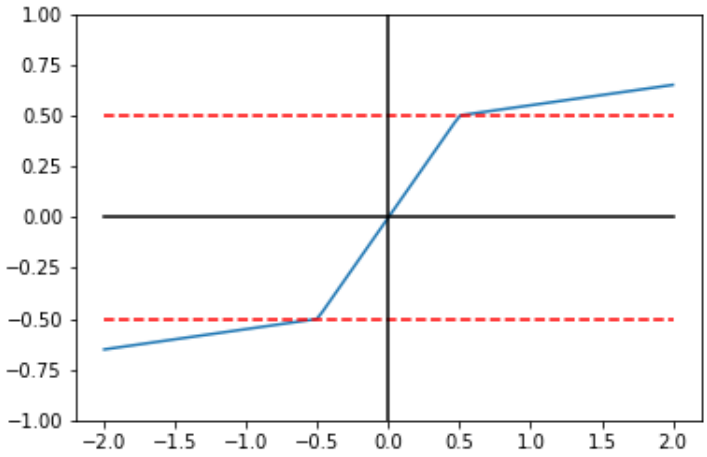

Figure 3.9: Illustration of adaptive clipping operation.

To make sure this operation is differentiable, we further reformulate Equation (3.12) as:

$$\begin{aligned}\tilde{\boldsymbol{x}}_{\boldsymbol{clip}} &= (1-\alpha) * (-\frac{1}{2}) + \alpha * min\{\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, -\frac{1}{2}\} \\ &\quad + (1-\alpha) * \frac{1}{2} + \alpha * max\{\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, \frac{1}{2}\} \\ &\quad + min\{max\{\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, -\frac{1}{2}\}, \frac{1}{2}\} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{clip}} &= \alpha * min\{\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, -\frac{1}{2}\} + \alpha * max\{\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, \frac{1}{2}\} \\ &\quad + min\{max\{\tilde{\boldsymbol{x}}_{\boldsymbol{conv}}, -\frac{1}{2}\}, \frac{1}{2}\}\end{aligned} \tag{3.13}$$

Learning the residual image eases the training process and leads to faster convergence, but since residual values are all close to zero this may induce the vanishing gradients problem. The initialization techniques used by the approaches described earlier are initializing all parameters to zero, or using Xavier initialization method [20] to draw each parameter from a random

zero-mean normal distribution with standard deviation of

$$std(W^i) = \sqrt{\frac{1}{N_{fan\_in}}} \tag{3.14}$$

where N_{in} is the number of incoming neurons.

As the initialization method significantly affects the performance of the CNN, we update the initialization method to the one described in [21]. Specifically, we initialize filter weights from a normal distribution with standard deviation of

$$std(W^i) = \sqrt{\frac{2}{N_{fan\_in}}} \tag{3.15}$$

He initialization [21] method is more suitable for this network than zero-initialization or Xavier initialization [20] for almost all convolutional layers are followed by a ReLU-like activation function, which drops half of the signal; hence, the number 2 in the numerator of the standard deviation is important in keeping the amplitude ratio of the output signal as close to 1 as possible.

3.3.7 SE-ARResNet Block

As we pointed out in Section 3.3.4, different channels in the DCT feature map include different frequency information. Considering the fact that the skip connection passes low-frequency information to later layers and the network only has to learn the residual image, we seek to exploit more useful information from the features maps for color image reconstruction. We utilize squeeze-and-excitation network [3] to exploit these inter-channel redundancies and assign weights to feature maps in all DCT, pixel and aggregation branches. Typical approaches in classification task take in the entire image

and resize them to a desired dimension before feeding into the network; however, this is not suitable for regression problems such as super-resolution or artifact removal as the interpolation operation at resizing time introduces additional loss. Therefore, as in super-resolution approaches, we break the input images down into patches of predefined sizes before feeding them into our network. Considering that the squeeze-and-excitation network performs well in the classification task by taking global information in the entire input image into consideration, we increase the size of input patches compared to previous super-resolution and artifact removal approaches such that more information can be considered by the squeeze-and-excitation block.

Following the success of ResNet [2] (Fig. 3.10(a)) in classification problems, many works have demonstrated that ResNet and its variant can be applied to regression problems such as image super-resolution with superior performance. SRResNet [22] (Fig. 3.10(b)) first removed the last ReLU activation layer as it drops half of the signal and is therefore not suitable for regression problems. EDSR [23] (Fig. 3.10(c)) later removed the batch normalization layer from SRResNet as batch normalization removes the range flexibility of features and requires too much memory. EDSR further modified the original ResNet architecture by adopting a residual scaling layer, which multiplies 0.1 to the output residual to make the model numerically stable.

Following these two architectures, we adopt a similar residual block with squeeze-and-excitation module tailored for compression artifact removal, shown in Fig. 3.10(d).

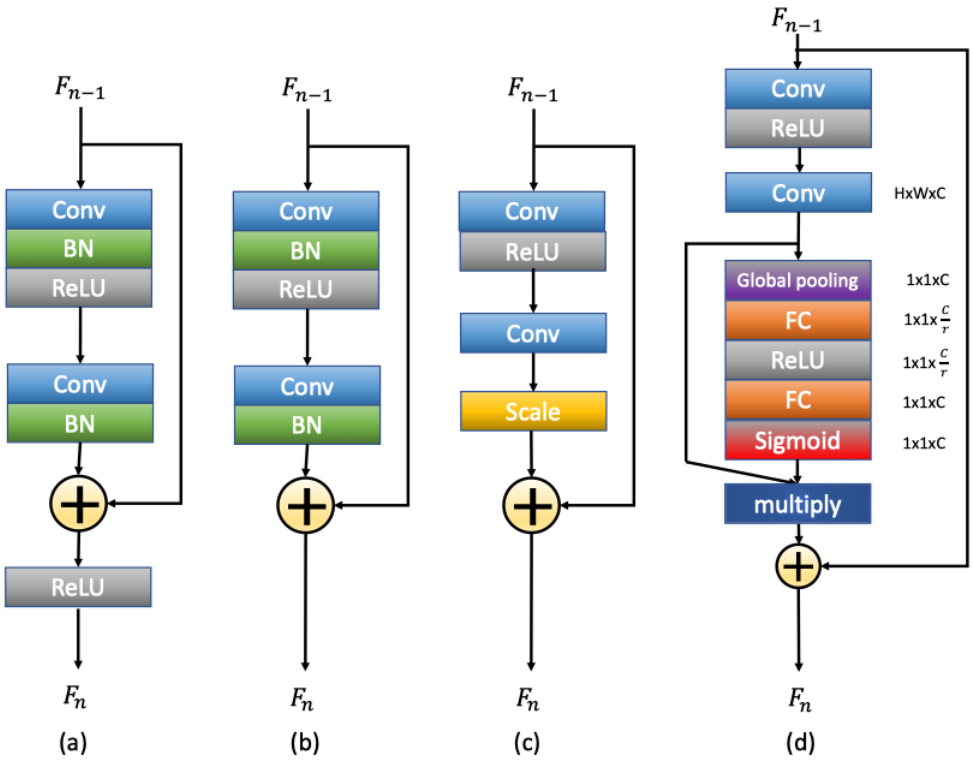

Figure 3.10: (a) Original ResNet. (b) SRResNet. (c) EDSR. (d) Proposed SE-ARResNet block.

In contrast to the EDSR in Fig. 3.10(c), to improve the representational power of our model, we replace the constant scaling layer with a squeeze-and-excitation block, which, instead of scaling the residual output from ResNet by a constant of 0.1, learns the scaling factor by exploiting model interdependencies between channels. We replace all convolutional layers in our model, i.e. $CONV_{D\{i\}}$ in DCT branch, $CONV_{P\{i\}}$ in pixel branch, and $CONV_{A\{i\}}$ in aggregation branch, with the proposed SE-ARResNet block. The final network architecture is shown in Fig. 3.11, where L_D, L_P and L_A denote the number of SE-ARResNet blocks in the DCT, pixel and aggregation branch.

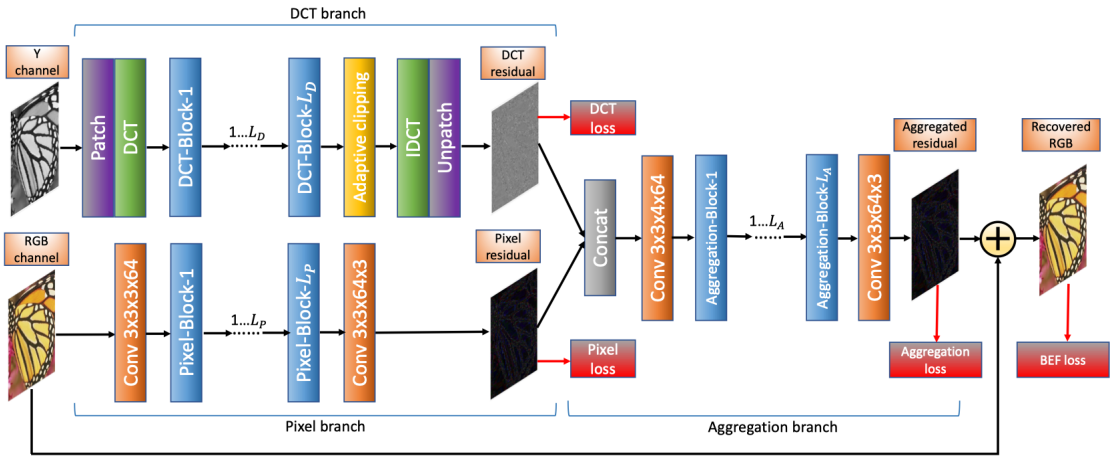

Figure 3.11: Proposed network architecture.

The DCT branch takes in the luminance channel of input patches and outputs a residual image in luminance channel as well. The pixel branch takes in the RGB image and produces the residual RGB image at the end. These two outputs are concatenated at the beginning of the aggregation branch, which in turns produces another RGB residual image. The input compressed RGB image is added to this final output to produce the recovered RGB image. An additional convolutional layer is placed at the beginning and end of both pixel and aggregation branches to adjust the feature map dimension accordingly.

For simplicity we denote the input RGB image as $\hat{\boldsymbol{x}}_{\boldsymbol{RGB}}$, the luminance image of the input patch as $\hat{\boldsymbol{x}}_{\boldsymbol{Y}}$, output residual from the pixel branch as $\hat{\boldsymbol{y}}_{\boldsymbol{P}}$, output residual from the DCT branch as $\hat{\boldsymbol{y}}_{\boldsymbol{D}}$, and output residual from the aggregation branch as $\hat{\boldsymbol{y}}_{\boldsymbol{A}}$, and recovered image as $\hat{\boldsymbol{y}}$. We also denote the groundtruth original RGB image as $\boldsymbol{x}_{\boldsymbol{RGB}}$, and the original luminance channel of the input image as $\boldsymbol{x}_{\boldsymbol{Y}}$. The recovered output image is therefore formulated as:

$$\hat{\boldsymbol{y}} = \boldsymbol{x}_{\boldsymbol{RGB}} + \hat{\boldsymbol{y}}_{\boldsymbol{A}} \tag{3.16}$$

Unlike VDSR [17] and DDCN [15] which formulate the loss function as the l2-norm between the groundtruth residual and the output residual from the model, we define a loss function that contains four terms: DCT loss, pixel loss, aggregation loss and BEF loss.

DCT loss directly models the difference between the groundtruth residual image in frequency domain and the output residual image of the DCT branch, as:

$$Loss_{DCT}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{DCT}(\hat{\boldsymbol{y}}_{\boldsymbol{D}}^{(\boldsymbol{i})} + \hat{\boldsymbol{x}}_{\boldsymbol{Y}}^{(\boldsymbol{i})}) - \boldsymbol{DCT}(\boldsymbol{x}_{\boldsymbol{Y}}^{(\boldsymbol{i})})\right\|_2^2 \tag{3.17}$$

Similarly, the pixel penalizes the difference between pixel branch output residual and the groundtruth RGB residuals in pixel domain:

$$Loss_{Pixel}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{y}}_{\boldsymbol{P}}^{(\boldsymbol{i})} - (\boldsymbol{x}_{\boldsymbol{RGB}}^{(\boldsymbol{i})} - \hat{\boldsymbol{x}}_{\boldsymbol{RGB}}^{(\boldsymbol{i})})\right\|_2^2 \tag{3.18}$$

The aggregation loss also accumulates the difference between aggregation branch output residual and the groundtruth RGB residuals in pixel domain:

$$Loss_{Aggregation}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{y}}_{\boldsymbol{A}}^{(\boldsymbol{i})} - (\boldsymbol{x}_{\boldsymbol{RGB}}^{(\boldsymbol{i})} - \hat{\boldsymbol{x}}_{\boldsymbol{RGB}}^{(\boldsymbol{i})})\right\|_2^2 \tag{3.19}$$

The BEF loss comes from the metric PSNR-B mentioned earlier in Section 1.5.3. It penalizes visible boundary artifacts in the aggregation output residual images where the differences in adjacent pixel intensities across DCT block boundaries are larger than those of pixel intensities that are not across DCT block boundaries. Formally, we define it as

$$Loss_{BEF}(\Theta) = \frac{1}{N}\sum_{i=1}^{N}\eta * [D_B(\hat{\boldsymbol{y}}^{(\boldsymbol{i})}) - D_B^C(\hat{\boldsymbol{y}}^{(\boldsymbol{i})})] \tag{3.20}$$

where η, D_B and D_B^C are defined in the same way as in Section 1.5.3.

Putting it together, the loss function that our model optimizes is:

$$
\begin{aligned}
Loss_{total}(\Theta) &= Loss_{DCT}(\Theta) + Loss_{Pixel}(\Theta) + Loss_{Aggregation}(\Theta) + Loss_{BEF}(\Theta) \\
&= \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{DCT}(\hat{\boldsymbol{y}}_{\boldsymbol{D}}^{(\boldsymbol{i})} + \hat{\boldsymbol{x}}_{\boldsymbol{Y}}^{(\boldsymbol{i})}) - \boldsymbol{DCT}(\boldsymbol{x}_{\boldsymbol{Y}}^{(\boldsymbol{i})})\right\|_2^2 \\
&+ \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{y}}_{\boldsymbol{P}}^{(\boldsymbol{i})} - (\boldsymbol{x}_{\boldsymbol{RGB}}^{(\boldsymbol{i})} - \hat{\boldsymbol{x}}_{\boldsymbol{RGB}}^{(\boldsymbol{i})})\right\|_2^2 \\
&+ \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{y}}_{\boldsymbol{A}}^{(\boldsymbol{i})} - (\boldsymbol{x}_{\boldsymbol{RGB}}^{(\boldsymbol{i})} - \hat{\boldsymbol{x}}_{\boldsymbol{RGB}}^{(\boldsymbol{i})})\right\|_2^2 \\
&+ \frac{1}{N}\sum_{i=1}^{N}\eta * [D_B(\hat{\boldsymbol{y}}^{(\boldsymbol{i})}) - D_B^C(\hat{\boldsymbol{y}}^{(\boldsymbol{i})})]
\end{aligned}
\tag{3.21}
$$

CHAPTER 4

EXPERIMENTS

4.1 Dataset

For training and testing dataset selection, we follow previous works in super-resolution and artifact removal and use the BSDS500 [24] as well as the LIVE1 dataset. BSDS500 has 500 images that are divided into 200 training images, 100 validation images and 200 testing images. LIVE1 dataset has another 29 testing images we use for testing in this project. We use the 200 training images in BSDS500 for training and the 100 validation images to validate the model. BSDS500 is a very large dataset and takes too much time to train; therefore, we also use the training dataset described in [25, 26], which has 91 color images and will be referred to as Set91, to perform some experiments and ablation studies. In this case the 14 images collected in [27] are used for testing. These training and testing images were carefully selected as they each demonstrate a variety of textures and edges that are important for image restoration. Data augmentation was adopted to increase the training set and improve generalization of our model. Specifically, we flip and rotate all training images and then extract smaller overlapping patches to feed into our networks. For SRCNN, ARCNN, VDSR and our proposed SE-ARResNet, we feed input patches of size 33, 32, 41 and 64 respectively into the network with a batch size of 64. In order to minimize degradation along patch boundaries, for SE-ARResNet we only crop out the center 48x48

pixels as output of the model for evaluation and reconstruction; therefore, at validation and testing time we extract patches at a stride of 48.

4.2 Implementation Details

In our implementation, we place 10 SE-ARResNet blocks as described in Section 3.3.7 in the DCT branch, pixel branch and aggregation branch respectively. The first convolutional layer in DCT branch has an input channel of 1 to match the input luminance image, and similarly the first convolutional layer in pixel branch has an input channel of 3 to match the input RGB image. The output feature maps of these two branches have 1 and 3 channels respectively to reconstruct the residual images. Consequently the first convolutional channel of the aggregation branch has an input channel of 4 to accommodate the concatenation of DCT and pixel branch output. All filter weights in convolutional and fully-connected layers are initialized by drawing from a truncated normal distribution with standard deviations specified using He initialization [21]. All remaining convolutional layers have 64 channels and stride 1. L_D, L_P and L_A are all set to 10, meaning ten SE-ARResNet blocks in each branch. In addition, both patch layer and unpatch layer have stride 8. This is to make sure that at patching time each 8×8 DCT block is reshaped into $1 \times 1 \times 64$ while at unpatching time each $1 \times 1 \times 64$ vector is reshaped back to a 8×8 block. The patch layer is implemented as a convolution while the unpatch layer as a transposed convolution.

4.2.1 Training Details

The α in Eq. (3.12) are initialized to 0.1 to each channel in the DCT feature map but updated as training proceeds. The reduction ratio r in the

squeeze-and-excitation block is set to 4 which is empirically the best choice. Following the success mentioned in [15], we use Adam optimizer with first and second momentum of 0.9 and 0.999 to limit each gradient update and prevent gradient explosion. In addition, the learning rate is initialized to 0.0001 but decreased by 10 every time the validation PSNR hits a plateau. We define this plateau as where the validation PSNR does not vary over 0.05dB in the last 5 epochs. Training is stopped after decreasing the learning rate 3 times. The entire training process takes slightly less than 7 hours on a GPU.

4.3 Results

Following previous work, we report PSNR and SSIM in Table 4.1 to demonstrate the reconstruction ability of each network. Although previous work only reported evaluations on luminance channel, we believe it is not sufficient since the luminance channel does not indicate the reconstruction ability of color components; therefore, we also report PSNR and SSIM of the reconstructed RGB images.

Table 4.1: Quantitative result on Set14 with QF=30. (Y) indicates only luminance channel is evaluated.

Metric	JPEG	SRCNN	ARCNN	VDSR	DDCN	Ours
PSNR	28.783	29.128	29.797	30.041	30.377	**30.543**
SSIM	0.943	0.947	0.952	0.955	0.957	**0.958**
PSNR(Y)	32.785	33.181	33.892	34.046	34.295	**34.380**
SSIM(Y)	0.905	0.910	0.918	0.920	0.922	**0.923**
PSNRB(Y)	32.152	32.859	33.746	33.930	34.136	**34.268**

One reason that networks targeted at super-resolution tasks are not suitable for suppressing compression artifacts is that they are not tailored to remove blocking and ringing artifacts. A sample residual output of VDSR

is shown in Fig. 4.1, and we can see that the network is not able to handle blocking artifacts well. Sample recovered color images from our proposed network are shown in Fig. 4.2 and Fig. 4.3 on page 57 and 58.

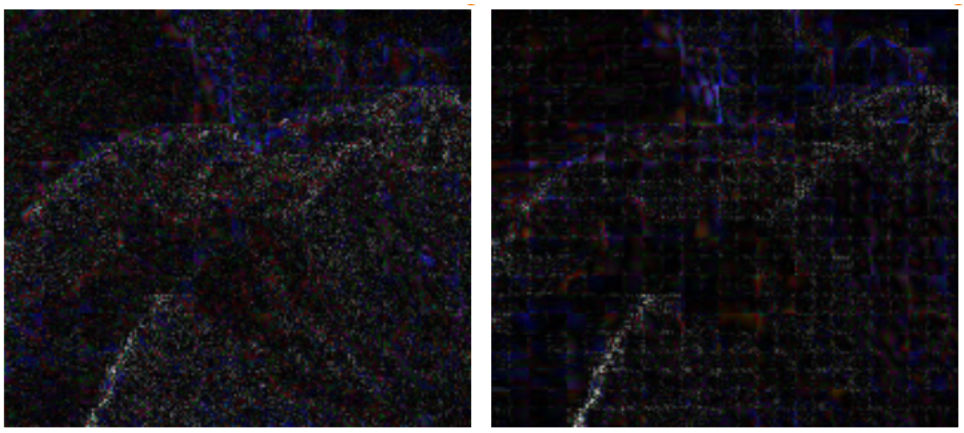

Figure 4.1: Left: An example groundtruth RGB residual image. Right: Output residual image from VDSR. VDSR fails to handle compression artifacts, specifically blocking artifacts in this example.

To quantitatively compare our model with previous approaches, we list the PSNR and SSIM of the luminance channel of reconstructed color images in Table 4.2 and Table 4.3. We also report PSNR-B of the luminance channel of the recovered images as blocking artifacts do not manifest differently in chrominance channels.

Table 4.2: Quantitative result of reconstruction of luminance channel on LIVE1 dataset.

Quality	Metric	JPEG	ARCNN	CAS_CNN	DMCNN	Ours
20	PSNR	30.62	31.78	31.70	**32.09**	32.03
	SSIM	0.868	0.890	0.895	**0.905**	0.892
	PSNR-B	27.57	30.69	30.88	31.32	**31.93**
10	PSNR	28.36	29.49	29.44	**29.73**	29.68
	SSIM	0.791	0.823	0.833	**0.842**	0.829
	PSNR-B	25.33	28.74	29.19	29.55	**29.62**

Table 4.3: Quantitative result of reconstruction of luminance channel on BSDS500 testing dataset.

Quality	Metric	JPEG	ARCNN	DDCN	DMCNN	Ours
20	PSNR	30.61	31.71	31.88	**31.98**	31.95
	SSIM	0.867	0.885	0.900	**0.904**	0.888
	PSNR-B	27.22	30.55	31.10	31.29	**31.81**
10	PSNR	28.39	29.45	29.59	**29.73**	29.65
	SSIM	0.810	0.834	0.838	**0.840**	0.835
	PSNR-B	25.10	28.73	29.18	29.33	**29.58**

As Table 4.1 indicates, the proposed dual-domain squeeze-and-excitation network recovers original color with comparable effectiveness but at much lower cost. Specifically, training this network takes less than 8 hours, while previous state-of-the-art models takes several days to converge. In addition, as we can see from Table 4.2 and Table 4.3, as well as Fig. 4.2 and Fig. 4.3, our models gives a higher PSNR-B, which indicates its ability to better remove blocking artifacts. This ability is also evidenced by the sample recovered images as blocking and banding artifacts are completely invisible, but high-frequency information such as sharp edges is not fully recovered.

4.3.1 Ablation Study

In this section we perform ablation studies on select key components of our model and show that these components do help improve the performance of our artifact removal model. Due to the limitations in resources, the ablation studies are conducted on a relatively small dataset, namely Set91, for faster convergence.

We first investigate whether replacing the conventional convolutional layers with residual blocks improves the model performance. In this experiment we replace convolutional layers in all three branches with residual blocks from

ResNet. Each residual block has the same structure as EDSR, as shown in Fig. 3.10(c), with scaling factor 0.1. All convolutional layers in residual blocks have 64 channels. We add 1 convolutional layer at the beginning and end of pixel and aggregation branches to change dimensionality accordingly. The two models are trained on images with $QF = 30$. Quantitative comparisons are given in Table 4.4. All metrics are evaluated on RGB colorspace.

Table 4.4: Study of residual blocks.

Network	PSNR	SSIM	PSNR(Y)	SSIM(Y)	PSNRB(Y)
JPEG	28.783	0.967	32.785	0.919	32.152
No residual model	30.308	0.974	34.252	0.927	34.168
Residual model	**30.405**	**0.974**	**34.350**	**0.927**	**34.251**

From Table 4.4 we can see that replacing the convolutional layers with residual blocks improves the model performance by around 0.1 dB.

Next we investigate the contribution of the squeeze-and-excitation module in our model. We replace the scaling layer in Fig. 3.10(c) with the squeeze-and-excitation (SE) module and keep input patch size the same. Effectively, we are learning the scaling factor from the SE module instead of using the constant 0.1. We set the reduction ratio r to 4, which is empirically found better among choices of 4, 8 and 16. This reduces the feature map from 64 channels to 16 channels in the squeezing phase of all SE modules. Again we train the two models on images with $QF = 30$ and show the evaluations in Table 4.5.

Table 4.5: Study of squeeze-and-excitation module.

Network	PSNR	SSIM	PSNR(Y)	SSIM(Y)	PSNRB(Y)
JPEG	28.783	0.967	32.785	0.919	32.152
No SE module	30.308	0.974	34.252	0.927	34.168
With SE module	**30.486**	**0.974**	**34.287**	**0.927**	**34.187**

Table 4.6: Study of larger input patch size.

Network	PSNR	SSIM	PSNR(Y)	SSIM(Y)	PSNRB(Y)
JPEG	28.783	0.967	32.785	0.919	32.152
Input 48x48	30.486	0.974	34.287	0.927	34.187
Input 64x64	**30.568**	**0.974**	**34.381**	**0.927**	**34.286**

The squeeze-and-excitation module works better by embedding global information. Since the model takes image patches as input and is not allowed to resize them, we increase the input patch size from 48x48 to 64x64, and the output size is increased from 32x32 to 48x48 correspondingly to avoid boundary effect. This increase results in additional improvement in performance, as shown in Table 4.6.

Evaluation results in Table 4.6 verify our proposition that using the squeeze-and-excitation module benefits from embedding more global information, but we do note that the improvement by adding squeeze-and-excitation networks to our model is not significant. To the best of our knowledge there are no existing works applying squeeze-and-excitation to regression problems such as image super-resolution, enhancement or restoration, and it is unclear whether this technique can be widely beneficial in these fields.

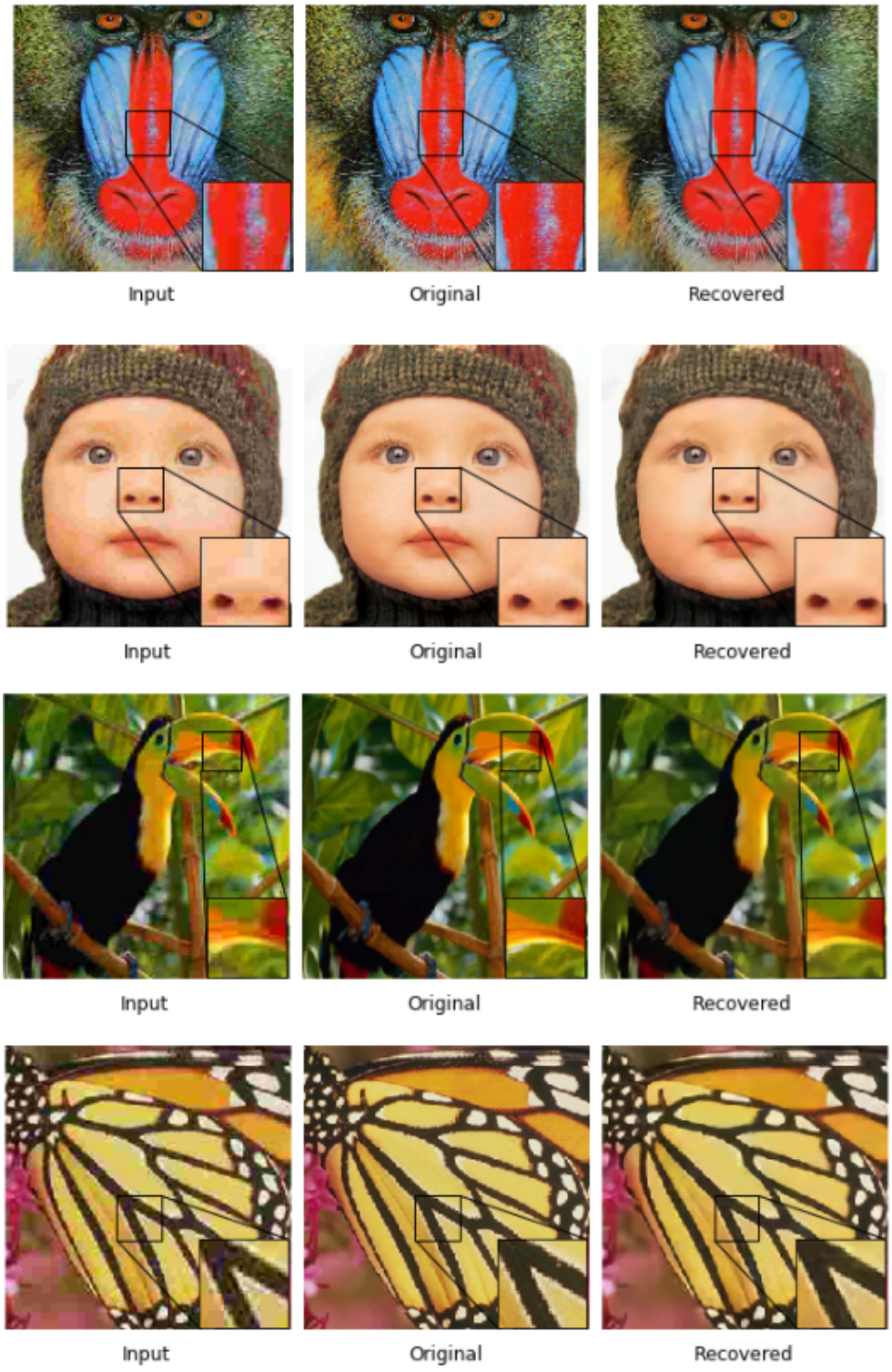

Figure 4.2: Sample images recovered by our proposed network. Left: Input compressed image. Middle: Original image. Right: Reconstructed image.

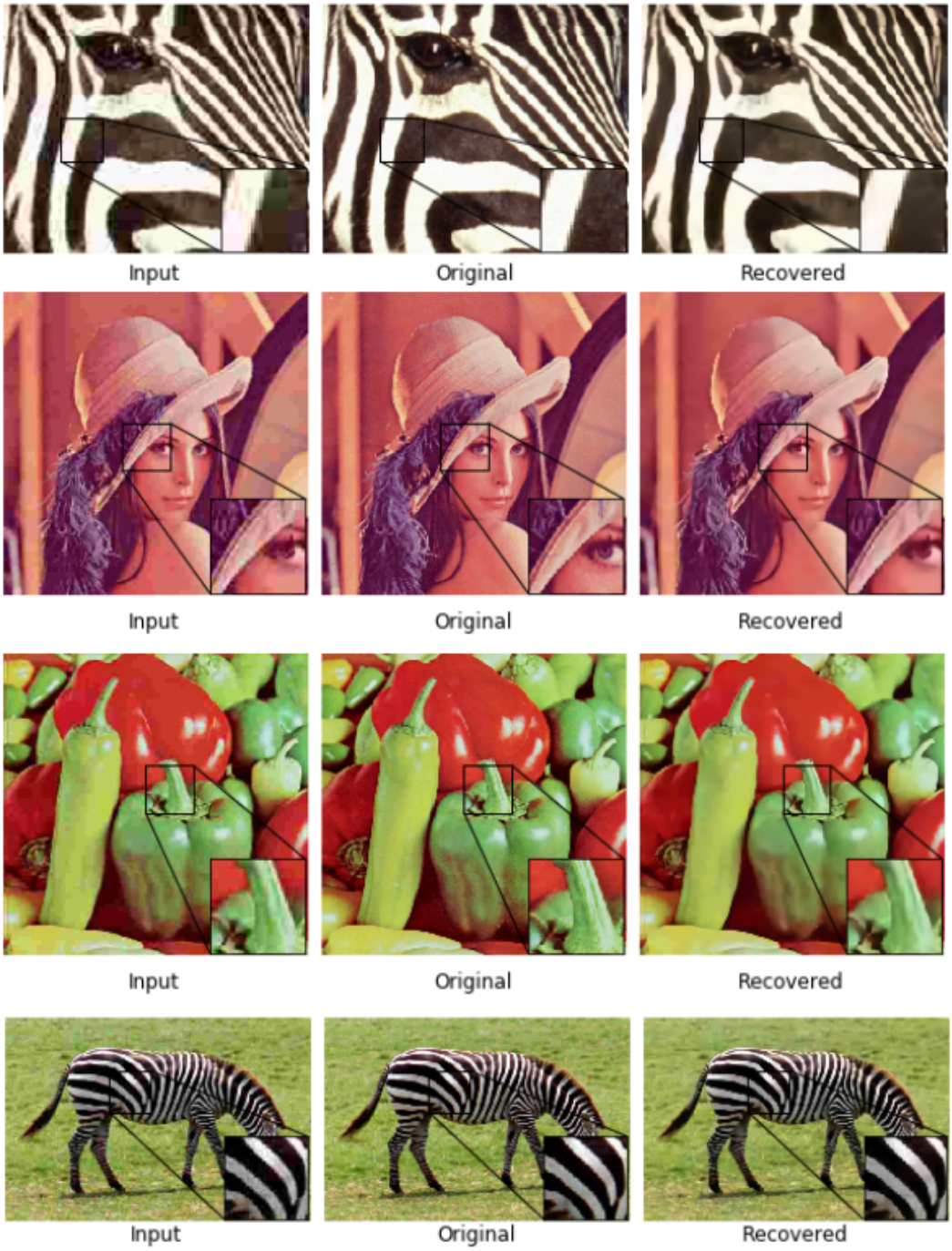

Figure 4.3: Sample images recovered by our proposed network. Left: Input compressed image. Middle: Original image. Right: Reconstructed image.

CHAPTER 5

CONCLUSIONS

In this thesis, we demonstrated that by leveraging prior knowledge of JPEG compression and applying a data-driven approach in both pixel and frequency domains, we are able to recover original color images from their compressed counterparts. The prior knowledge allows us to constrain quantization errors within DCT domain and recover high-frequency information. We also incorporated residual blocks and squeeze-and-excitation modules in our approach to exploit inter-channel redundancies and grow the network deeper to improve its representational power. From both quantitative and qualitative evaluations we can see that the proposed network is able to reconstruct satisfying artifact-free images, and can be trained faster than other networks.

One potential improvement of the proposed network is to further recover high-frequency components. In our experiments we noticed that compression artifacts are generally easier to remove but high-frequency information is still hard to recover. We believe training on a larger and more representative dataset will help tackle this challenge. Another possible technique that may help recover high-frequency information is to add additional reconstruction in wavelet domain, and it would be very important to find the right balance between improved reconstruction and additional computations.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257 pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *CVPR09*, 2009.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli et al., "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[6] C. Yim and A. C. Bovik, "Quality assessment of deblocked images," *IEEE Transactions on Image Processing*, vol. 20, no. 1, pp. 88–98, 2011.

[7] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," *CoRR*, vol. abs/1504.06993, 2015. [Online]. Available: http://arxiv.org/abs/1504.06993

[8] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transactions on Circuits and Sstems for Video Technology*, vol. 13, no. 7, pp. 614–619, 2003.

[9] H. C. Reeve and J. S. Lim, "Reduction of blocking effects in image coding," *Optical Engineering*, vol. 23, no. 1, p. 230134, 1984.

[10] A.-C. Liew and H. Yan, "Blocking artifacts suppression in block-coded images using overcomplete wavelet representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 4, pp. 450–461, 2004.

[11] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Projection-based spatially adaptive reconstruction of block-transform compressed images," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 896–908, 1995.

[12] J. Jancsary, S. Nowozin, and C. Rother, "Loss-specific training of non-parametric image restoration models: A new state of the art," in *European Conference on Computer Vision*. Springer, 2012, pp. 112–125.

[13] C. Jung, L. Jiao, H. Qi, and T. Sun, "Image deblocking via sparse representation," *Signal Processing: Image Communication*, vol. 27, no. 6, pp. 663–677, 2012.

[14] X. Liu, X. Wu, J. Zhou, and D. Zhao, "Data-driven sparsity-based restoration of jpeg-compressed images in dual transform-pixel domain," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5171–5178.

[15] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *European Conference on Computer Vision*. Springer, 2016, pp. 628–644.

[16] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision*. Springer, 2014, pp. 184–199.

[17] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[18] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 576–584.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[20] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4681–4690.

[23] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 136–144.

[24] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

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

[26] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[27] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International Conference on Curves and Surfaces*. Springer, 2010, pp. 711–730.