The present work was submitted to the
Chair of Computer Science 13 (Computer Vision)
Faculty of Mathematics, Computer Science and Natural Sciences
Prof. Dr. Bastian Leibe

Master Thesis

Flexible Blind JPEG Artifacts Removal

presented by

Jiaxi Jiang
Student ID: 373947
2021-04-19

First examiner:	Prof. Dr. Bastian Leibe (RWTH Aachen)
Second examiner:	Prof. Dr. Luc Van Gool (ETH Zurich)
Advised by:	Dr. Kai Zhang (ETH Zurich)
	Dr. Radu Timofte (ETH Zurich)

Eidesstattliche Versicherung

Jiaxi Jiang	373947
Name	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel

Flexible Blind JPEG Artifacts Removal

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 2021-04-19	
Ort, Datum	Unterschrift

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten dementsprechend.

Die vorstehende Belehrung habe ich zur Kentnis genommen:

Aachen, 2021-04-19	
Ort, Datum	Unterschrift

Abstract

Training a single deep blind model to handle different quality factors for JPEG image artifacts removal has been attracting considerable attention due to its convenience for practical usage. However, existing deep blind methods usually directly reconstruct the restored image without predicting the quality factor, thus lacking the flexibility to control the output as the non-blind methods and failing to fully take advantage of the quality factor information for better reconstruction. As a result, they do not perform well in real applications as the real images could be compressed more than once. To remedy this, in this thesis, we propose a flexible deep blind JPEG artifacts removal network, namely FBAR, that can not only use quality factor during training for better restoration but also predict the adjustable quality factor to flexibly control the trade-off between artifacts removal and details preservation. To improve the flexibility, FBAR decouples the adjustable quality factor from the JPEG image via a decoupler module and then embeds the predicted quality factor into the subsequent reconstructor module through a quality factor attention block for flexible control. To improve the practicability, a practical degradation model which involves double JPEG compression is proposed for data synthesis. Extensive experiments on single JPEG compressed images, the more general double compressed JPEG images and real-world JPEG images demonstrate that the proposed FBAR achieves favorable performance against state-of-the-art methods in terms of quantitative metrics and visual quality.

Acknowledgements

This thesis is done during my research stay at Computer Vision Lab, ETH Zurich, within the framework of IDEA League, an alliance among five leading European universities of technology.

I would like to convey my deep appreciation to my advisors Dr. Kai Zhang and Dr. Radu Timofte for their constant guidance and encouragement throughout the thesis, and for always steering me in the right direction. I also would like to express sincere gratitude to Prof. Luc Van Gool for his kind support and supervision, and for making it possible for me to work on the project. I would like to thank all members in Computer Vision Lab at ETH Zurich for their help and technical support for my thesis.

I am extremely grateful to my supervisor Prof. Bastian Leibe for continuous help and inspiration during my studies at RWTH Aachen University, and for offering me the opportunity to explore the incredible world of computer vision. I would like to thank Jonathon Luiten and other members in Computer Vision Group at RWTH Aachen University for their nice tutoring and discussion.

Finally, I would like to express my sincerest appreciation to my parents and grandparents for their endless support and encouragement in my life. And to Xin, for always being by my side.

Contents

1 Introduction

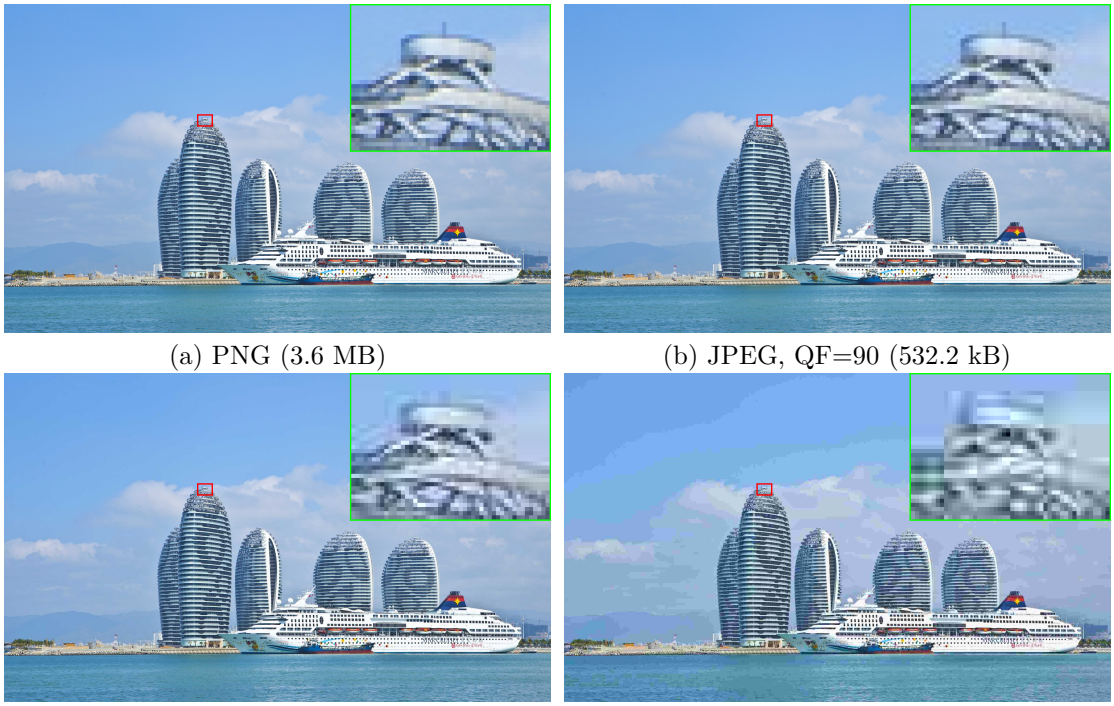
1.1. Motivation

With the widespread availability of the Internet and the development of affordable, high-quality digital cameras, lossy image compression technique which uses inexact approximations but reduced information for representing the encoded image is indispensable and inevitable for saving storage size and bandwidth. While introduced three decades ago, JPEG compression [Wal92] is still one of the most widely-used image compression formats due to its simplicity and fast encoding/decoding speeds. JPEG compression splits an image into 8×8 blocks and applies discrete cosine transform (DCT) to each block. The DCT coefficients are then divided by quantization table and rounded to the nearest integer. The elements in the quantization table control the compression ratio and the rounding operation is the only lossy operation in the whole process. The quantization table is often represented by an integer called quality factor (QF) ranging from 0 to 100, where a lower quality factor means less storage size but more lost information. An example of JPEG compression can be found in Figure 1.1.

As a lossy compression algorithm, JPEG compression usually introduces annoying blocky artifacts, which deteriorates the visual quality and hinders the performance of subsequent low-level vision processing such as super-resolution and high-level vision processing such as object detection. Inspired by the success of deep neural networks (DNNs) for image classification [KSH12, SZ15], researchers began to resort to DNNs for JPEG artifacts removal and have achieved notable academic success.

However, existing methods for JPEG artifacts removal generally have four limitations in real applications:

(1) While JPEG compression can adjust the quality factor for a flexible trade-off between storage size and visual quality, most existing DNNs based methods [CHB17, CP16, DDCLT15, LZZ$^+$18, ZLL$^+$19] trained a specific model for each quality factor, lacking the flexibility to learn a single model to handle different



(a) PNG (3.6 MB) (b) JPEG, QF=90 (532.2 kB)

(c) JPEG, QF=50 (208.9 kB) (d) JPEG, QF=10 (89.0 kB)

Figure 1.1.: **An illustration to show the effects of JPEG compression.** We can control the tradeoff between saving storage size and preserving image quality by setting different quality factors. Image from DIV2K Dataset [TAVG+17].

JPEG quality factors. It is expensive and impractical to train and deploy so many models for real use.

(2) DCT-based methods [ELDS20, GC16, ZYHL18] need to obtain the DCT coefficients or quantization table as input, which is only stored in JPEG format. In real world, it is common that a JPEG image is resaved as another format such as PNG, making the required prior information not accessible. Besides, when images are compressed multiple times, the stored quantization table only reflects the most recent compression information. As a result, a potential gap between different quality factors along the degradation history will severely reduce the restoration performance.

(3) To solve the first problem, recent works [ELDS20, FZW+19, ZZC+17] resort to training a single model for a large range of quality factors. However, these blind methods can only provide a deterministic reconstruction result for each input, ignoring the need for user-preference. Actually, subjective visual perception is more important in real image restoration.

(4) Existing methods are all trained with synthetic images, which assume that the low-quality images are only compressed once. However, most images from the Internet are compressed many times. Despite the progresses for real recompressed

images, e.g., from Twitter [DDCLT15, FZW+19], a detailed and complete study on double JPEG artifacts removal is lacking.

As the results of current state-of-the-art methods on real JPEG images are often not satisfactory, a study to overcome the fourth problem would be a significant step towards real image restoration. We propose that this discrepancy is in part due to unrealistic synthetic training data. Many classic algorithms generalize poorly to real data due to assumptions that the image is compressed only once without considering the multiple steps of a typical image compression pipeline. One approach to ameliorate the mismatch between synthetic training data and real images is to collect images that have been spread many times on the Internet. However, it is impractical to also get their original ones. Given original images, collecting repeatedly transmitted images is as difficult as getting a specified drift bottle. Acquiring these image pairs is expensive and time-consuming, which is exacerbated by the large amounts of training data required to prevent overfitting when training neural networks. Furthermore, because different websites and software employ different compression algorithms, only getting from some of them is not enough. On the other hand, the real degradation model is often unknown, so it is almost impossible to include every possibility in the synthetic training set. Therefore, a controllable flexible model is well-understood and can be leveraged to deal with images with different degradation.

1.2. Contributions

To tackle the above problems, we design a flexible blind artifacts removal network, namely FBAR, for real JPEG image restoration. Our FBAR is a single model that can deal with JPEG images with different quality factors. In addition, FBAR can work independent of the image formats, as it directly processes images in pixel-domain, without the need to access meta information from the input images. By further decoupling the latent quality factor from the input JPEG image, we can use this important parameter to guide the artifacts removal process. More importantly, as a controllable variable with clear physical meaning, the predicted quality factor can be adjusted via interactive selection to balance artifacts removal and details preservation. To show the effectiveness of the proposed FBAR, we provide a detailed study on double JPEG compression for handling real JPEG images with severe degradation. The experiments show that some types of recompression are actually equivalent to single JPEG compression in terms of artifacts removal. On the other hand, when the 8×8 blocks of double JPEG compression are not aligned, and the first quality factor QF1 $\leq$ QF2, existing blind methods will fail to work. However, our quality factor predictor can help to explain the behavior of current blind methods under unseen scenarios. We provide comprehensive empirical evidence showing that blind methods work are easy to be mislead by the unseen compound artifacts, resulting in a unpleasant reconstructed output. By correcting the predicted quality factor, FBAR instead

can boost the performance of complex double JPEG images. To achieve a fully blind model, we further propose a simple but effective dominant quality factor prediction algorithm that can find the dominant quality factor that brings the main compression artifacts, by utilizing the properties of JPEG images.

To summarize, the main contributions of this thesis are:

(1) A flexible blind JPEG artifacts removal network (FBAR) is proposed. FBAR can predict the latent quality factor to guide the JPEG image restoration. More importantly, the predicted quality factor can be adjusted by users to achieve a balance between artifacts removal and details preservation.

(2) We perform a thorough analysis of images with double JPEG compression to take a step towards real JPEG images. To the best of our knowledge, this is the first attempt to handle double non-aligned JPEG compression. It is our hope that the community will gradually begin to consider this more challenging and realistic scenario.

(3) We demonstrate the effectiveness of proposed FBAR on synthetic and real JPEG images with complex degradation settings. Our proposed FBAR provides a useful solution for practical applications.

1.3. Thesis Organization

The thesis is organized as follows:

- Chapter 1 introduces the background, motivation, goals, and contributions of the thesis.
- Chapter 2 will summarize the previous work on the topics of JPEG artifacts removal, double JPEG compression, and flexible image restoration.
- Chapter 3 will review the necessary theoretical background in image restoration and deep learning for our work.
- Chapter 4 will present our proposed novel solution towards real JPEG artifacts removal. We also address the importance of flexibility by showing the limitations of existing methods on double JPEG compressed images, and present the dominant quality factor prediction algorithm which help to get a fully blind model.
- Chapter 5 will show the experiments conducted on the synthetic single, double, triple JPEG images and real-world JPEG images. We will also present the application of our method in other vision tasks.
- Chapter 6 will give concluding remarks, summarize the major contributions of our work, and present the future outlooks.

2 Related Work

In this chapter, we will briefly review the previous work related to this thesis.

First, we will introduce the progress that has been achieved for the tasks of JPEG artifacts removal in Section 2.1. Since deep learning-based methods have shown surpassing performance in several image restoration tasks lead to a promising outlook, we will not consider traditional methods in this section.

Afterwards, we will present the work related to double JPEG compression in Section 2.2, which is an active research area in image forensics but has not been studied in the context of image restoration yet.

Since one of our novelties is the flexibility in image restoration, we will also introduce the related flexible image generation tasks by which we are inspired in Section 2.3.

2.1. JPEG Artifacts Removal Networks

Learning-based methods have made notable progress in JPEG artifacts removal in the past few years. Dong *et al.* [DDCLT15] first introduced deep learning to remove JPEG artifacts, inspired by the success of super-resolution network [DLHT14]. Chen and Pock [CP16] proposed a dynamic nonlinear reaction diffusion model with time-dependent parameters for a variety of image restoration, including JPEG deblocking. In [CHB17], a 12-layer deep convolutional network with hierarchical skip connections was presented and trained with a multi-scale loss function. Zhang *et al.* [ZZC+17] employed batch normalization [IS15] and residual learning [HZRS16a] strategies to speed up the training process and boost the performance on general blind image restoration task. A wavelet transform based network was presented in [LZZ+18] as the generalization of dilated convolution [YK16] and subsampling, leading to a large improvement. Fu *et al.* [FZW+19] proposed a deep convolutional sparse coding network that combines model-based methods with deep learning. To extract the long-range dependencies between pixels, a residual non-local attention network is designed in [ZLL+19],

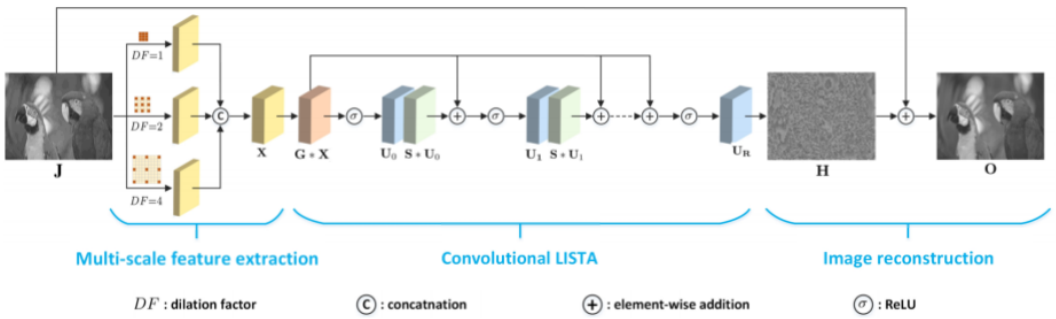


Figure 2.1.: **The architecture of DCSC.** DCSC combines traditional optimization methods with deep learning, and is made up of three components: multi-scale feature extraction, convolutinoal learned iterative shrinkage threshold algorithm, and image reconstruction. Figure adapted from [FZW+19].

which resulted in better representational ability. Besides, dual-domain convolutional network based methods [GC16,KSC20,ZYHL18,ZCT+19] were proposed to take the advantages of redundancies on both pixel and DCT domains. Recently, Ehrlich *et al.* [ELDS20] trained their networks with the utilization of quantization matrix as prior information, which allows a single model to correct artifacts at any quality factor and achieved state-of-the-art results.

However, most existing methods train a model for each quality factor, making it impossible in the real application. Besides, DCT-based approaches need the DCT coefficients or quantization table as the input, but such information is only stored in JPEG format and only records the most recent compression. Furthermore, existing blind methods which train a single model for multiple quality factors only gives a deterministic restored result, without considering the need for user preference. In addition, all of the current JPEG artifacts removal methods assume the JPEG image is only compressed once, which is far from the possible severe degradation in the wild.

Figure 2.1 illustrates the restoration procedure of DCSC [FZW+19], which is specially designed for JPEG artifacts removal using deep convolutional sparse coding, a method that combines traditional methods and deep learning together. Traditional methods for reducing JPEG artifacts are usually based on model-based optimization, which is well explainable in physical meaning but time-consuming during inference. On the contrary, learning-based methods enjoy fast inference speed, but have less explainability. DCSC is set up based on the framework of learned iterative shrinkage threshold algorithm (LISTA) for convolutional sparse coding (CSC). Besides, dilated convolutions are employed for multi-scale feature extraction, which can increase the receptive fields to handle multiple JPEG quality factors without introducing extra parameters. The final output is obtained by adding the corrupted input to the learned residual image, which is similar to [ZZC+17].

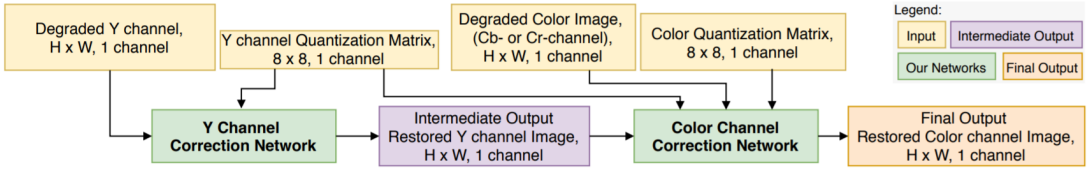


Figure 2.2.: **The architecture of QGAC.** Given a JPEG image, the pixel information and quantization matrix of Y channel are extracted as input to firstly get the restored Y channel. Next, this intermediate output, together with degraded Cb and Cr channels as well as quantization matrix of all channels, are fed into the color channel correction network to get the final restored color image. Figure from [ELDS20].

We also show the architecture of QGAC [ELDS20] in Figure 2.2. While existing methods only consider restoration of the luminance Y channel, QGAC firstly restores the Y channel image as an intermediate step towards color JPEG image restoration. We should note that in this architecture, the quantization matrix is a required input. However, this information is only stored in JPEG format and only reflects the most recent compression information. In this thesis, we will show the limitation of such DCT-based methods in the real application.

2.2. Double JPEG Compression

Double JPEG compression has been studied in the area of image forensics for a long time, as detection of double compression can provide important clues for recovery of image processing history. Fu *et al.* [FSS07] showed that if an image has been JPEG compressed only once, then the first digits of the quantized JPEG coefficients follow a Benford-like logarithmic law. In [BCS10,BP11,CH11,LQHQ07], double JPEG compression was classified into two cases: aligned and non-aligned. Chen *et al.* [CH11] formulated the periodic characteristics of JPEG images in both spatial and DCT domains and showed that such periodic characteristics will be changed after recompression. Recently, learning-based methods [BBB+17, PCAL18, WZ16] were proposed to detect double JPEG compression. Besides, estimation of the first quantization table of JPEG images is also a challenging problem and studied in both aligned [GPBB14, PBPG14, XYL+17, YHN+16] and non-aligned cases [BP12, DO18, YWQZ20]. However, these methods focus on analyzing the DCT coefficients, which are only stored in JPEG format. Besides, the research on double JPEG compression restoration is still lacking.

In fact, the non-aligned double JPEG compression is very common in daily life. For example, a non-aligned double compressed JPEG image is easy to generate by cropping out a region of interest from a JPEG image, and then save the region also in JPEG format, as shown in Figure 2.3. Besides, most social network websites always downsample the uploaded images and then do JPEG compression. If the uploaded image is a JPEG image, then a non-aligned double compression occurs.

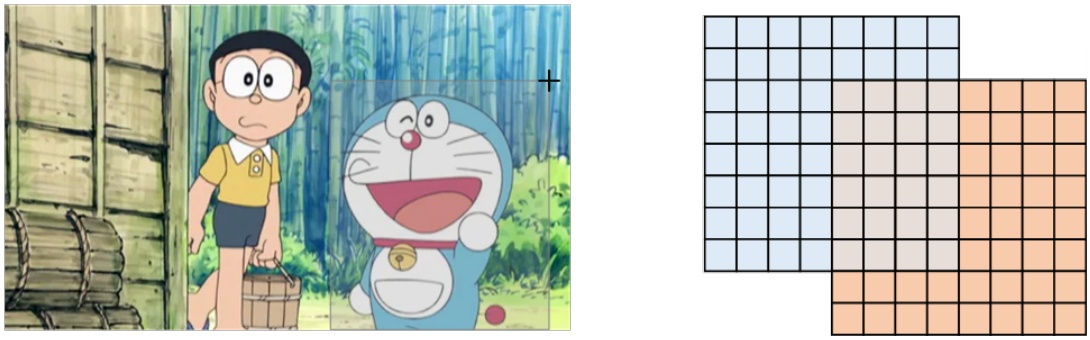


Figure 2.3.: **An example of non-aligned double JPEG compression. Left:** A common practical scenario when we want to crop out a region of interest from a JPEG image. If we save the region of interest also as JPEG, then a double JPEG compression occurs. **Right:** In this scenario, if the 8×8 blocks of the newly saved image from the region of interest are not aligned with the blocks of the original JPEG image, then this is non-aligned double JPEG compression.

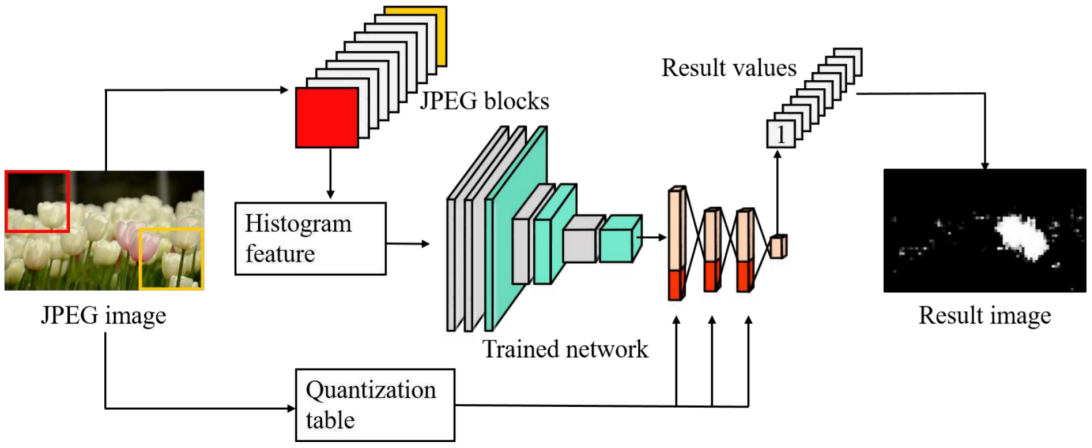


Figure 2.4.: **A learning-based double JPEG detection method proposed in [PCAL18].** The manipulated regions can be identified by detecting the double JPEG blocks. Figure from [PCAL18].

The learning-based double JPEG detection method for image tampering proposed in [PCAL18] is shown in Figure 2.4. The input of the neural network is the histogram feature extracted from the DCT coefficients of 8×8 JPEG blocks within an image, while the output is a 2×1 vector representing the classification result (single/double JPEG). By sliding window over the whole image, manipulated regions can be detected effectively. This image tampering detection is based on that single and double JPEG compressed image often have different artifacts. In our thesis, we propose that aligned double JPEG compression and non-aligned double JPEG compression with QF1 > QF2 are equivalent to single JPEG compression, which will also be verified by this method.

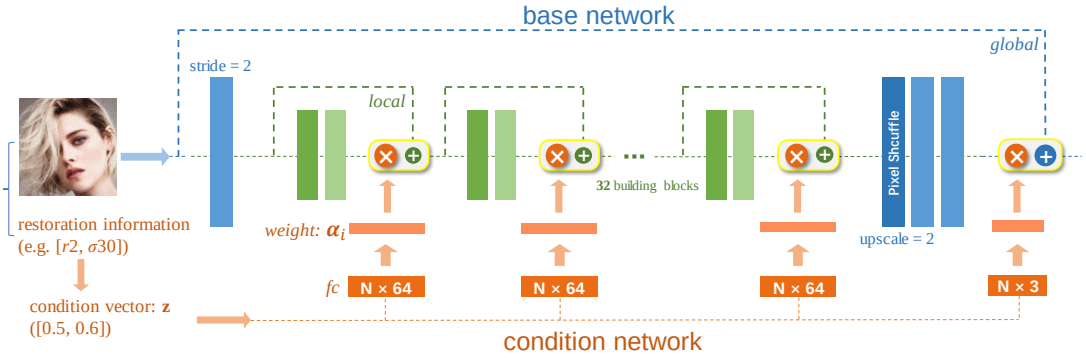


Figure 2.5.: **The framework of CResMD for controllable image restoration.** Figure from [HDQ20].

2.3. Flexible Image Restoration

Flexible image generation based on the conditional variable has drawn much attention, e.g., in text-to-image generation [LQLT19, RAY+16, XZH+18, ZXL+17] and facial attribute editing [CNL+20, CCK+18, HZK+19, LDX+19]. However, these methods can not be directly adopted in image restoration. Zhang *et al.* [ZZZ18a] proposed to take a tunable noise level map as the input to handle noise on different levels. In [ZZZ18b], a PCA-based dimensionality stretching of the degradation parameters was proposed to take blur kernel and noise level as input for super-resolution. Wang *et al.* [WGTY19] proposed a novel controllable framework for interactive image restoration. He *et al.* [HDQ20] focused on the images with multiple degradations and added the multi-dimensional degradation information as input, as illustrated in Figure 2.5. However, these methods usually assume that the controllable variable is provided, but such information is almost unknown in real applications. This encourages us to work towards a flexible blind solution.

Inspired by the ideas in image attribute editing and flexible image restoration, we regard the quality factor of JPEG images as the image attribute in this thesis. The quality factor can be utilized to flexibly control the desired output to make a trade-off between preserving texture details and removing JPEG artifacts. More importantly, our method is both blind and non-blind, which can automatically provide an ideal restored result and control the output through interactive selection.

3 Preliminaries

In this chapter, background knowledge about JPEG compression algorithm and the general problem formulation in image restoration are stated, and a selection of specific deep learning approaches which are related to our proposed method is presented.

We will start by introducing the encoding and decoding steps of JPEG compression algorithm in Section 3.1. Following this, we will give the general problem formulation of image restoration tasks, including JPEG artifacts removal and other common tasks in Section 3.2. Finally, relevant deep learning frameworks and techniques are explained in Section 3.3.

3.1. JPEG Compression Algorithm

JPEG compression [Wal92] is the name given to an image compression algorithm developed by the Joint Photographic Experts Group in 1992. While introduced three decades ago, JPEG compression is still the most widely used image compression standard in the world due to its simplicity and fast encoding/decoding speeds. It is also the most widely used digital image format. The degree of compression can be adjusted, allowing a controllable trade-off between saving storage size and preserving image quality. In this section, we will introduce the basic steps of encoding and decoding of the JPEG compression algorithm respectively. An overview of JPEG encoding and decoding procedures can be seen in Figure 3.1.

3.1.1. Encoding

The main steps of encoding in JPEG compression are color space transformation, downsampling, block splitting, discrete cosine transform, quantization, and entropy coding. We introduce these steps in detail as follows.

Color Space Transformation: We take the color images as an example because grayscale images are regarded as Y' channel images and follow the same subsequent procedure. Given a color image, it is firstly converted from RGB space to

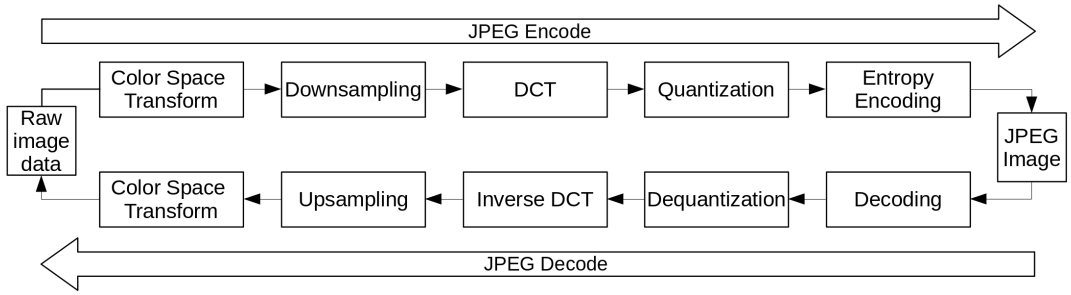


Figure 3.1.: **The general procedure of JPEG algorithm.** JPEG compression algorithm consists of encoding and decoding steps.

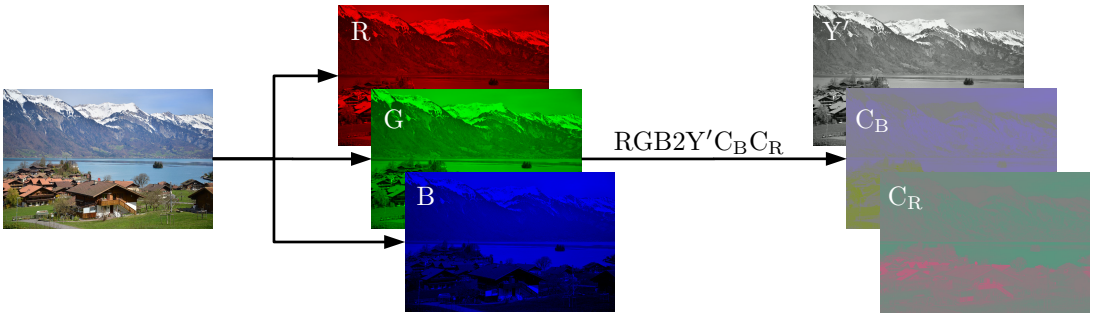


Figure 3.2.: **Color space transformation from** RGB **to** $Y'C_BC_R$ **space.** This procedure is based on that human eyes are more sensitive to the variations of brightness than color. After the transformation, each channel is processed separately.

$Y'C_BC_R$ space, consisting of one luma component (Y'), representing light intensity, and two chroma components (C_B and C_R), representing color. Please note that luma (Y') is different from luminance (Y) because of the consideration of the gamma correction. According to JPEG encoding standard in MATLAB, the transformation is given by:

$$\begin{aligned} Y' &= \frac{1}{255}(65.481 \cdot R + 128.553 \cdot G + 24.966 \cdot B) + 16 \\ C_B &= \frac{1}{255}(-37.797 \cdot R - 74.203 \cdot G + 112.0 \cdot B) + 128 \\ C_R &= \frac{1}{255}(112.0 \cdot R - 93.786 \cdot G - 18.214 \cdot B) + 128 \end{aligned} \tag{3.1}$$

$Y'C_BC_R$ space allows confining the brightness information, which is more important to the visual quality, to a single channel so that greater compression can

be done without a significant drop on perceptual quality of the image. This step is based on the fact that perception principle of the human visual system.

Downsampling: The step of downsampling is also called chroma subsampling, which reduces the spatial resolution of the C_B and C_R channels. Typically, the downsampling ratio is set as 4:4:4 (no chroma subsampling), 4:2:2 (halve the horizontal chroma resolution), 4:2:0 (have both the horizontal and vertical chroma resolution). This step is based on the fact that the human visual system is much more sensitive to variations in brightness than color. Next, the three channels are processed independently in a similar way.

Block Splitting: After downsampling, each channel is split into 8×8 blocks. If the size of horizontal or vertical direction is not a multiple of 8, then the remaining incomplete blocks must be filled with values. Filling the edges with constant values will usually generate rining artifacts along the border, so more advanced border filling techniques can be applied to reduce such artifacts.

Discrete Cosine Transform: Discrete cosine transform (DCT) is applied to each 8×8 blocks and converts the image representation from the spatial domain to the frequency domain. More specifically, each entry of the block is firstly centered on zero. Then two-dimensional DCT is applied for each block, which is given by:

$$D_{i,j} = \frac{1}{4}\alpha(i)\alpha(j)\sum_{x=0}^{7}\sum_{y=0}^{7} P_{x,y}\cos\left[\frac{(2x+1)i\pi}{16}\right]\cos\left[\frac{(2y+1)j\pi}{16}\right] \tag{3.2}$$

where

- $\alpha(i) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & \text{otherwise} \end{cases}$ is the normalizing scale factor.
- The integers i and j are the horizontal and vertical spatial frequency with $0 \leq i, j \leq 7$.
- $P_{x,y}$ is the zero-centered pixel value at coordinates at (x, y).
- $D_{i,j}$ is the DCT coefficients at coordinates (i, j).

The top-left corner entry of the obtained DCT is called DC coefficient and has a rather large magnitude. The remaining 63 coefficients are called AC coefficients. In this way, the final signal can be placed in a corner making it easy to be compressed efficiently in the stage of entropy.

Quantization: The entries of each block after DCT transform represent the frequency. The goal of the quantization step is to reduce the high-frequency components to save storage space. This is because human eyes are good at

distinguishing slight differences in brightness over a large area but not so sensitive to the variation of high-frequency brightness.

This step is quite straightforward. We only need to divide each entry of the DCT coefficients by a constant for that component and then rounding to the nearest integer. The constant value is called the quantization step, and each component of the 8×8 DCT coefficients within a block corresponds to a quantization step. The quantization step is computed with:

$$D^q_{i,j} = \text{round}(\frac{D_{i,j}}{Q_{i,j}}), 0 \leq i, j \leq 7 \tag{3.3}$$

where $D^q_{i,j}$ and $Q^q_{i,j}$ are the quantized DCT coefficient and the quantization step at the coordinate (i, j).
All the quantization steps form a 8×8 matrix called quantization table (or quantization matrix). After the quantization step, most higher frequency components are rounded to zero, and the rest components take much less storage space. Please note that the rounding operation is the only lossy operation in the whole process other than chroma subsampling.

Depending on the size of the quantization steps, more or less information is lost in this step. In most software, users can define the degree of the JPEG compression by setting the quality factor, which is an integer ranging from 0 to 100 and associated with a quantization table. It should be pointed out that quantization is the only step where users have an influence on the result.

Entropy Coding: Entropy coding is a technique of lossless data compression. In this step, the quantized DCT coefficients are arranged in a zigzag order, which groups similar frequencies together, as illustrated in Figure 3.3. Zeros are coded by a number representing the length of zero, and the remaining components are coded using Huffman encoding.

3.1.2. Decoding

The decoding procedure largely follows the reverse steps of the encoding procedure. First, by taking the element-wise product of the DCT coefficient matrix and quantization matrix, the original DCT coefficient is reconstructed, with the bottom-right components as zeros. Next, two-dimensional inverse DCT is applied to the reconstructed DCT coefficients, given by:

$$P_{x,y} = \frac{1}{4}\alpha(i)\alpha(j)\sum_{x=0}^{7}\sum_{y=0}^{7} D_{x,y}\cos\left[\frac{(2x+1)i\pi}{16}\right]\cos\left[\frac{(2y+1)j\pi}{16}\right] \tag{3.4}$$

where

- $\alpha(i)$is the normalizing scale factor defined as above.

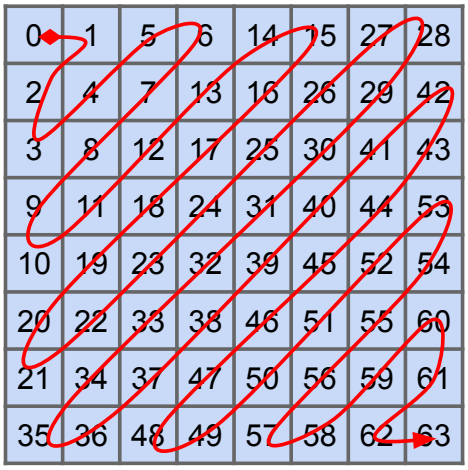


Figure 3.3.: **Zigzag ordering for the arrangement of DCT coefficients.** In this way the similar frequencies are grouped together.

- The integers x and y are the pixel values in horizontal and vertical direction with $0 \leq x, y \leq 7$.
- $D_{i,j}$ is the reconstructed DCT coefficients at coordinates (i, j).
- $P_{x,y}$ is the reconstructed zero-centered pixel value at coordinates at (x, y).

Then the reconstructed zero-centered pixel values are rounded to integer values and added with 128 to each entry, to get the reconstructed images in $\mathrm{Y'C_BC_R}$ space. The decoding process may generate values outside the range of [0, 255], so the output should be clipped to prevent overflow. Finally, the image is converted back to RGB color space, which is computed by:

$$\begin{aligned} \mathrm{R} &= 255 \cdot (0.00456621 \cdot \mathrm{Y'} + 0 \cdot \mathrm{C_B} + 0.00625893 \cdot \mathrm{C_R}) - 222.921 \\ \mathrm{G} &= 255 \cdot (0.00456621 \cdot \mathrm{Y'} - 0.00153632 \cdot \mathrm{C_B} - 0.00318811 \cdot \mathrm{C_R}) + 135.576 \\ \mathrm{B} &= 255 \cdot (0.00456621 \cdot \mathrm{Y'} + 0.00791071 \cdot \mathrm{C_B} - 0 \cdot \mathrm{C_R}) - 276.836 \end{aligned} \tag{3.5}$$

3.2. Image Restoration

Image restoration is a classical low-level computer vision problem that has been studied for a long time due to its highly practical value in various real applications ranging from medical imaging, film archival, forensic science to consumer photography. In this section, we will give the problem formulation of image restoration, introduce the common image restoration tasks, and tell the difference between blind and non-blind methods.

3.2.1. Problem Formulation

In general, the purpose of image restoration is to take an observed corrupted low-quality image and estimate the clean original image. Different from image enhancement which aims to improve an image subjectively, image restoration removes distortion from images to reconstruct the original one, which is an objective process.

The overall steps of image restoration are first to model the degradation and then apply the inverse process. So designing a degradation model which can accurately describe the degradation history is a key step in image restoration. Generally, the degradation model can be formulated as:

$$\mathbf{y} = \mathbf{H}[\mathbf{x}] + \mathbf{n} \tag{3.6}$$

where

- x is the original clean image.
- y is the degraded observation.
- $\mathbf{H}[\cdot]$ is a degradation function.
- $\mathbf{n}$ is additive noise, e.g. white Gaussian noise with standard deviation σ.

Image restoration is an ill-posed inverse problem, so regularization needs to be imposed to constrain the solution. Most traditional methods solve this problem from a Bayesian perspective on the framework of Maximum A Posterior (MAP):

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \log p(\mathbf{x}) \tag{3.7}$$

where $\hat{\mathbf{x}}$ is the estimated reconstructed result, $\log p(\mathbf{y}|\mathbf{x})$ is the log-likelihood of degraded observation $\mathbf{y}, \log p(\mathbf{x})$ represents the prior of $\mathbf{x}$ and is independent of $\mathbf{y}$. More formally, Equation 3.7 can be reformulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{H}[\mathbf{x}]\|^2 + \lambda\Phi(\mathbf{x}) \tag{3.8}$$

where the objective is to minimize an energy function made up of a data fidelity term $\frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$ and a regularization term $\Phi(\mathbf{x})$. The parameter λ controls the trade-off between them. The fidelity term guarantees that the solution is in line with the degradation model, while the regularization term ensures the desired property of the output.

The main challenge in image restoration is that much information has been lost during the degradation, making it a highly ill-posed inverse problem. To get a good reconstructed result, prior knowledge is required to provide additional information. Therefore, it is significant to model the prior of clean images properly.

Besides, by specifying different degradation functions, we can correspondingly define and solve different image restoration tasks. Four classical image restoration would be image denoising when $\mathbf{H}$ is an identity matrix, image deblurring when $\mathbf{H}$ is a blurring operator, image super-resolution when $\mathbf{H}$ is a composite operator of blurring and down-sampling, JPEG artifacts removal when $\mathbf{H}$ is JPEG compression. We briefly introduce these tasks as follows.

Denoising: The goal of image denoising is to recover a clean, high-quality image from its noisy observation. The noise in an image may come from the transmission of electronic components, the limitations of imaging systems, or other influences and disturbances from the environment. A commonly adopted image degradation model in the research community for the denoising tasks is written as:

$$\mathbf{y} = \mathbf{x} + \mathbf{n} \tag{3.9}$$

Most denoising methods tackle the additive white Gaussian noise (AWGN) case, which assumes n to be the independent identically distributed Gaussian noise with zero mean and standard variance σ. The noise can also be of other types, such as Poisson noise or pepper and salt noise.

Deblurring: Deblurring is the process of removing blurring artifacts from blurred observations to get clear and sharp images. This kind of degradation may come from the low resolution of the camera, out of focus, or high-speed motion. The degradation model in image deblurring is usually written as:

$$\mathbf{y} = \mathbf{x} \otimes \mathbf{k} + \mathbf{n} \tag{3.10}$$

where $\otimes$ denotes the 2D convolution operator, and k is the blur kernel. In imaging systems, the blur kernel is the shape of an idealized point imaged through the optical system and called the point spread function (PSF). Traditionally, image deblurring problems are solved through inverse filtering in the frequency domain. The image deblurring problem can be split into two distinct problems: recovering the PSF and recovering the original image using a known PSF. While blind deconvolution methods try to recover the PSF, non-blind methods focused on robust deconvolution based on a known PSF.

Super-Resolution: Single image super-resolution (SISR) is also a classical image restoration problem. The goal is to estimate the clean, high-resolution counterpart of a given low-resolution image. Since imaging systems have limited resolution, SISR has wide applications in various areas such as face recognition, medical imaging, and remote sensing. The general degradation model for SISR is given by

$$\mathbf{y} = (\mathbf{x} \otimes \mathbf{k}) \downarrow_s + \mathbf{n} \tag{3.11}$$

where $\mathbf{x} \otimes \mathbf{k}$ represents the convolution between blur kernel k and high-resolution image x. $\downarrow_s$ is a subsequent downsampling operation with scale factor s. $\mathbf{n}$ is

additive white Gaussian noise with noise level σ. When the scale factor $s = 1$, SISR becomes deblurring and denoising problems.

JPEG Artifacts Removal: The focus of our work is JPEG artifacts removal. JPEG artifacts are introduced during the quantization step in the encoding procedure in JPEG compression algorithm, as we have discussed in Section 3.1. Compared with the other three above image restoration tasks, the mathematical degradation model of JPEG compression is highly nonlinear and thus more complicated. Besides, as JPEG is both the most widely used image compression algorithm and the most widely used image format, JPEG artifacts are actually more common than Gaussian noises in the real wold. In the research community, the degradation caused by JPEG compression is usually given by:

$$\mathbf{y} = \mathrm{JPEG}(\mathbf{x}, \mathrm{QF}) \tag{3.12}$$

where the quality factor QF controls the degree of degradation.

This degradation model assumes that the corrupted JPEG image is compressed only once. However, in the real world, most JPEG images are compressed many times, and existing methods often fail to work on those real JPEG images. Designing a more realistic degradation model considering multiple JPEG compression to generate synthetic training pairs can be a solution. However, the degradation history of real images is always unknown, so the synthetic data do not necessarily describe the real degradation accurately. Therefore, in this thesis, we mainly focus on solving the real JPEG problems using the same degradation model as existing methods, proving a solution to real image restoration problems under limited training data.

Besides, the degradation model of JPEG compression can be combined with other tasks such as deblurring and SISR, which would be more practical in the real world.

3.2.2. Blind and Non-Blind Methods

The concepts of "blind" and "non-blind" are very common in image restoration, so it is necessary to explain the difference between them. According to whether the degradation information is available during the inference, image restoration methods can be classified into blind methods (available) and non-blind methods (not available). More specifically, this degradation information is represented by e.g. noise level in image denoising, blur kernel in image deblurring, and quality factor in JPEG artifacts removal.

In the real world, the degradation history of corrupted images is often unknown, so it is natural to think about designing a blind model trained with synthetic data with various degradation types for better real application. However, when the degradation is complex, e.g. motion blur, the performance would deteriorate seriously. This is because a low-quality image may correspond to different high-low quality images, which aggravates the pixel-wise average problem [LTH+17,

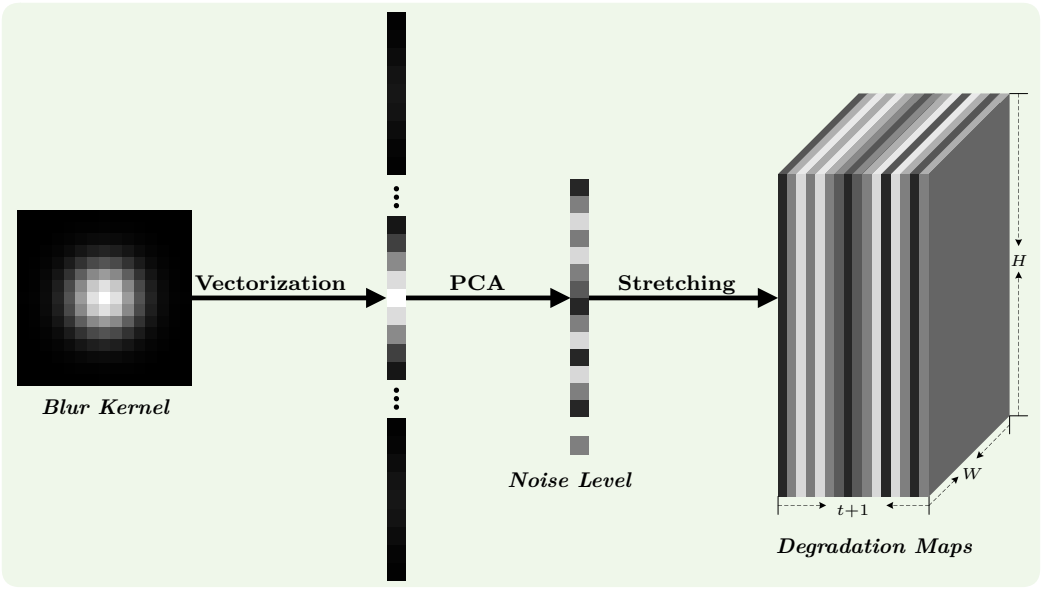


Figure 3.4.: **The dimensionality stretching strategy proposed in SRMD.** For a $W \times H$ low quality image, the vectorized blur kernel is first projected onto a t-dimensional linear space and then stretched into the $W \times H \times (t+1)$ degradation map with the noise level. Figure from [ZZZ18b].

ZZZ18b, ZZZ19], leading to an over-smoothed result. Besides, when combined with degradation prior as input, non-blind methods can be adjustable to control the trade-off between removing noise and preserving detail, which is promising for real noise removal. Furthermore, non-blind methods can be plugged into the variable splitting algorithms for general image restoration, including image deblurring, image super-resolution, and image inpainting.

However, it is not easy to directly feed both the degradation prior and low-quality image into learning-based methods, as they usually have different dimensions. To tackle this problem, FFDNet [ZZZ18a] stretched the noise level σ into a noise level map with all elements set as σ. This noise level map can also be further extended to degradation maps with multiple channels for more general noise removal problems. For the SISR task, SRMD [ZZZ18b] proposed a dimensionality stretching strategy which can combine the blur kernel with noise level into degradation maps. As illustrated in Figure 3.4, the blur kernel is vectorized into a vector and then projected onto a linear space by Principal Component Analysis (PCA). The next step is similar to FFDNet [ZZZ18a], where the concatenated low dimensional vector and the noise level are stretched into degradation maps.

The main limitation of non-blind methods lies in assuming that the degradation prior is known because most real corrupted images do not provide such information. A possible solution is to combine the non-blind methods with degradation estimation, e.g. noise level and blur kernel estimation. However, most degra-

dation parameter estimation methods are specially designed for a specific model and contain more or less errors. On the contrary, Non-blind methods are usually trained with true degradation history, which is often different from the estimated degradation. During inference, such a mismatch may lead to a large drop in performance. Another possible solution is to set up a series of possible degradation information to generate different results for a given corrupted image. Then users can choose the desired output according to their preference. However, generating multiple images will cost much more time.

In this thesis, we present a novel method that can predict the quality factor, which is the latent degradation parameter of a JPEG image, and use it to guide the restoration procedure. The quality factor prediction is jointly trained with image reconstruction, which solves the mismatch problem mentioned above. Besides, by setting different quality factors in an interactive way, the need for user preference is also considered. Since the quality factor is only fed into the decoder part, the deep image features extracted by the encoder is fixed, which save much time during interactive selections.

3.3. Deep Learning

The focus of our work is to develop a powerful JPEG artifacts removal model using deep learning. Therefore, we will introduce the basic knowledge of encoder-decoder architectures, residual learning, and attention mechanism, upon which our proposed method is built.

3.3.1. Encoder-Decoder Architectures

Autoencoder: Autoencoder is a neural network that is trained to learn efficient data representation in an unsupervised way. Figure 3.5 illustrates a general layout of an autoencoder, which maps an input $\mathbf{x}$ to a reconstructed output $\mathbf{x}'$ through an internal representation or code $\mathbf{z}$. It should be noted that the goal of autoencoders is not to learn to copy the input perfectly, which can be done by simply setting $\mathbf{x}'$ as $\mathbf{x}$. Instead, autoencoders usually learn useful properties of the input data by forcing them to learn certain properties of the training data. One way to extract such useful properties is to constrain the feature representation $\mathbf{z}$ to have a smaller dimension than $\mathbf{x}$. An autoencoder whose dimension in the bottleneck layer is less than that of the input is called an undercomplete autoencoder. Learning an undercomplete representation forces the autoencoder to extract the most salient features of the input data.

The autoencoder can be trained simply with $L2$ loss function, which is written as:

$$L(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2 \tag{3.13}$$

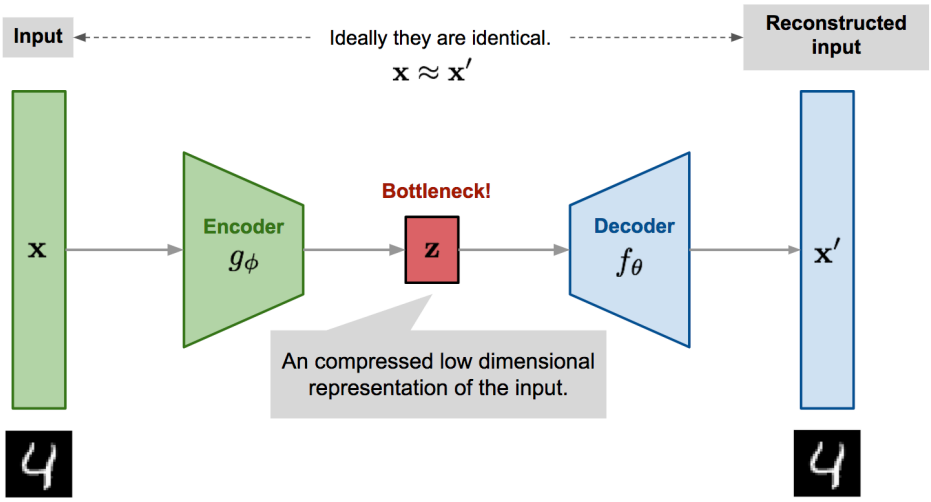


Figure 3.5.: **A schematic illustration of Autoencoder.** The autoencoder has two components: the encoder g_ϕ which maps $\mathbf{x}$ to $\mathbf{z}$, and the decoder f_θ, which maps $\mathbf{z}$ to an internal representation or code $\mathbf{x}'$.[1]

After training, the decoder part can be thrown away. In addition to dimension reduction, the encoder can be used to initialize a supervised model. Autoencoders were already around for decades but initially were using linear mappings, fully connected networks. If a linear decoder is used and the loss function is $L2$, an autoencoder learns the same subspace as a PCA would do. With nonlinear encoder and decoder functions, autoencoders can learn a more powerful nonlinear generalization of PCA.

However, if the encoder and decoder are equipped with too much capacity, the autoencoder can potentially learn an identity function, only copying input data without extracting useful information about the dataset. A similar problem occurs if the bottleneck representation has larger dimension than (overcomplete autoencoders), or equal to, the input. Various techniques have been proposed to prevent autoencoders from learning the identity function and to improve their ability to capture richer representations.

One solution of them is called regularized autoencoders, which force the model to learn other properties besides copying input. In this way we do not need to reduce the model capacity or code size. Regularized autoencoders can be nonlinear and overcomplete but still learn useful information about the dataset. Classical strategies for regularizing an autoencoder include denoising autoencoder [VLBM08] (change error term of the cost function), sparse autoencoder (add a sparse penalty to the cost function), and contractive autoencoder [RVM+11] (keep learned representation staying in a contractive space). In addtion to the regularized autoencoders, two generative modeling methods related to autoencoders but do not require regularization are variational autoencoders [KW14]

[1]Image source: https://lilianweng.github.io/lil-log/

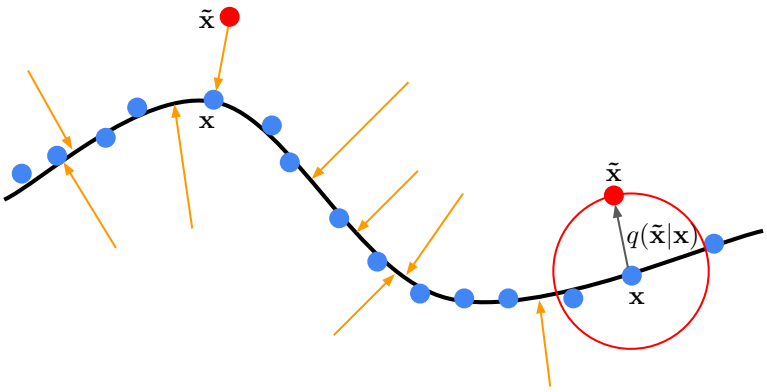


Figure 3.6.: **DAE explained by manifold learning.** Training data x (•) are lying near a low-dimensional manifold illustrated with the bold curve. Intermediate representation can be explained as a coordinate system for data points on the manifold. Noisy data $\tilde{x}$ (•) obtained by applying corruption process $q(\tilde{\mathbf{x}}|\mathbf{x})$ will be farther from the manifold.

and generative stochastic networks [BLAY14], which are trained to approximately maximize the probability of the training data.

Denoising Autoencoder: Since our proposed method is more related to

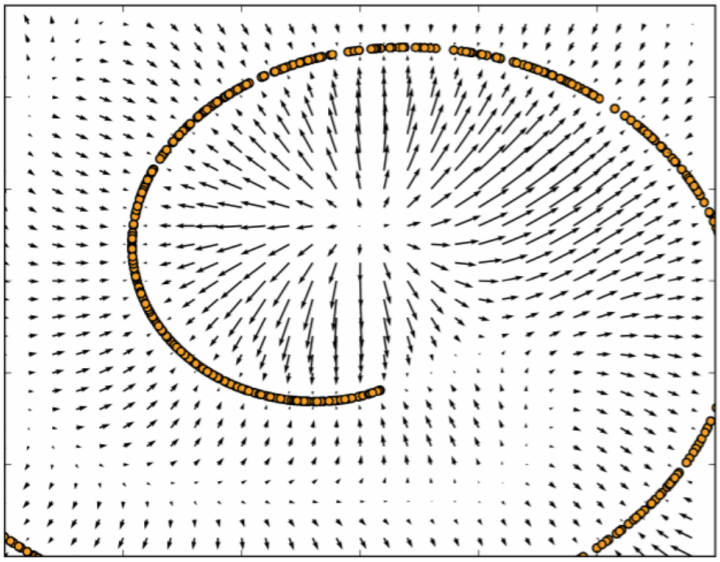


Figure 3.7.: **Vector field learned by a denoising autoencoder around the manifold**. Figure from [AB14].

denoising autoencoder, we introduce it here with more details. The denoising

autoencoder [VLBM08] (DAE) is an autoencoder, whose input is corrupted data and output is original clean data. DAE modifies the loss function in Equation 3.13 to:

$$L(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\tilde{\mathbf{x}}^{(i)})))^2 \tag{3.14}$$

where the input $\tilde{\mathbf{x}}$ is the corrupted noisy copy of original data $\mathbf{x}$. The denoising autoencoder can be explained from manifold learning. DAE assumes that natural data actually lies in a low-dimensional manifold of the high-dimensional space of input data $\mathbf{x}$. As illustrated in Figure 3.6, the intermediate representation $\mathbf{z} = g_\phi(\mathbf{x})$ is interpreted in [VLBM08] as a coordinate system for points on the manifold. The goal of training a denoising autoencoder to map a noisy data point $\tilde{x}$ back to the original clean data point x. A further explanation is shown inFigure 3.7, where the vector field learned by the DAE is illustrated. The vector field has zeros at both maxima and minima of the estimated density function on the manifolds. The autoencoder can map data points with low probability to higher probability reconstructions. When the probability is maximal, the reconstruction is more accurate, then the arrows shrink.

U-Net: U-Net [RFB15] is a classical successful encoder-decoder architecture which is developed from autoencoders. U-Net was originally designed for medical image segmentation but immediately also widely applied to other various tasks, including image restoration.

U-Net is a symmetric architecture and skips the connections between the up-sampling and down-sampling path using a concatenation operator. The architecture of U-Net is illustrated in Figure 3.8. Like most encoder-decoder architectures, U-Net has three parts: encoder, bottleneck, and decoder. The encoder part consists of four contraction blocks. Each block is made up of two 3×3 convolutions, with each convolution followed by a 2×2 max pooling operation with stride 2 for downsampling. The number of feature channels gets double after each down-sampling operation. The bottleneck part uses two 3×3 convolutions and 2×2 up convolution. The decoder part consists of several expansion blocks, with each block passing the input, a concatenation with the correspondingly cropped feature map from the contracting path, to two 3×3 convolutions and a 2×2 upsampling operation that halves the number of feature channels. Finally, 1×1 convolution is employed to adapt the number of feature channels to the same as the number of desired output channels.

The success of U-Net mainly lies in that it can combine the low-level image appearance information and high-level abstract information. The high-level abstract information can provide the contextual semantic information of the segmentation target in the entire image, which can be understood as a feature that reflects the relationship between the object and its environment. In contrast, the low-level high-resolution information is directly transferred from the encoder to the de-

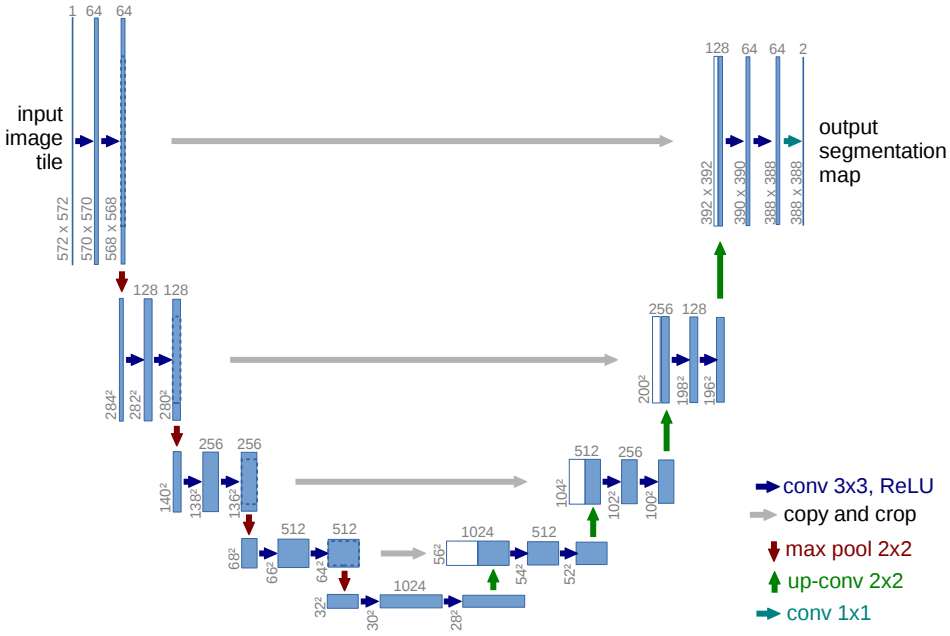


Figure 3.8.: **The architecture of U-Net**. Figure from [RFB15].

coder at the same height after concatenating operation, providing more refined features for segmentation, such as gradients.

3.3.2. Residual Learning

In the deep network model with deep levels, in addition to the problem of gradient diffusion, there is a problem of degradation. Batch Normalization is an effective method to solve the gradient diffusion problem [IS15]. The so-called degradation problem is that as the depth increases, the network accuracy reaches saturation and then drops rapidly, as is shown in Figure 3.9. In [HZRS16a], Residual Networks (ResNet) was used to solve the degradation problem. The main feature of ResNet is the cross-layer connection, which passes the input across layers and adds the result of the convolution by introducing shortcut connections. There is only one sampling layer in ResNet, which is connected behind the last convolutional layer. ResNet enables the underlying network to be fully trained and the accuracy is significantly improved as the depth deepens. Using 152-layer ResNet for the LSVRC-15 image classification competition, it won the first place. In [HZRS16a], an attempt was also made to set the depth of ResNet to 1000 and validate the model in the CIFAR-10 image processing data set.

The original residual unit in [HZRS16a] made the computation:

$$\mathbf{y}_l = h(\mathbf{x}_l) + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) \tag{3.15}$$

$$\mathbf{x}_{l+1} = f(\mathbf{y}_l) \tag{3.16}$$

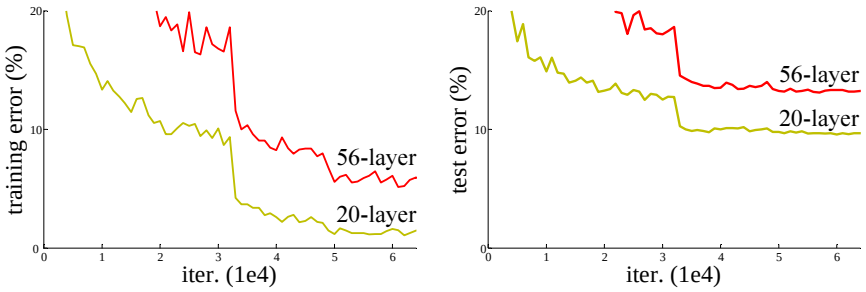


Figure 3.9.: **Training and test error on CIFAR-10.** The deeper network has higher training error and test error, which is not as expected. Figure from [HZRS16a].

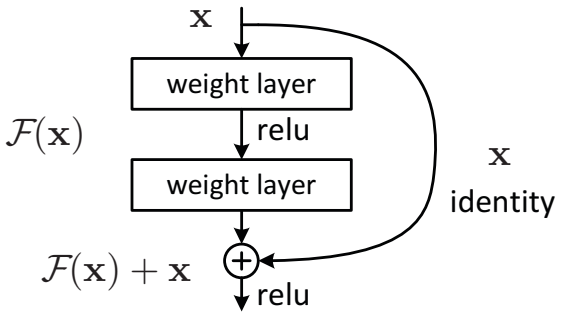


Figure 3.10.: **The building block of ResNet.** Figure from [HZRS16a].

Where $\mathbf{x}_l$ is the input feature to the l-th Residual Unit and $\mathcal{W}_l$ are the weights including biases related to the l-th Residual Unit. The function f is an operation after addition (in [HZRS16a] is set as $ReLU$). If the function h and the function f are both set as identity mappings: $h(\mathbf{x}_l) = \mathbf{x}_l$ and $\mathbf{x}_{l+1} \equiv \mathbf{y}_l$, we can then get the relationship between adjacent layers:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l) \tag{3.17}$$

If we perform the computation recursively, we will obtain:

$$\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i) \tag{3.18}$$

There are three main properties of ResNet, according to [HZRS16b]:

- Any deep unit $\mathbf{x}_L$ can be represented by any shallower unit $\mathbf{x}_l$ plus residual functions $\sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)$.
- Any deep unit $\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)$ can be decomposed of $\mathbf{x}_l$ and the outputs of the residual functions between the layers l and $L-1$.

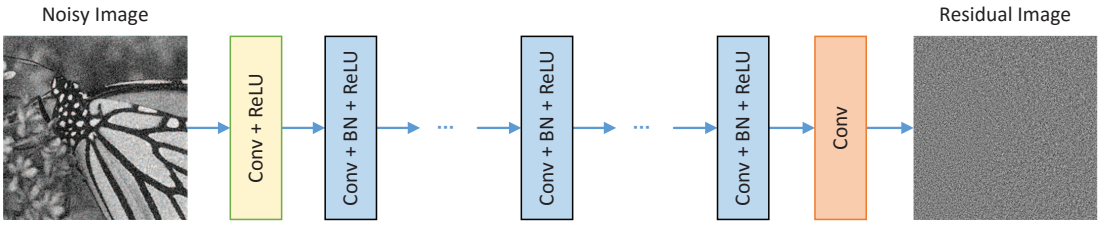


Figure 3.11.: **The architecture of DnCNN.** Figure from [ZZC+17].

- Equation 3.18 implies good back-propagation properties.

The third property is very meaningful. As analyzed in [HZRS16b], given loss function denoted as $\mathcal{E}$, we have:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=1}^{L-1}\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)) \tag{3.19}$$

The equation shows that the gradient can be decomposed into a term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ that propagates information directly without concerning any weight layers and another term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}(\frac{\partial}{\partial \mathbf{x}_l}\sum_{i=1}^{L-1}\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i))$ that propagates through the weight layers. The term of $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ ensures that information is directly propagated back to any shallower unit l. This equation also suggests that it is impossible for the gradient $\frac{\partial}{\partial \mathbf{x}_l}$ to vanish, because for a mini-batch the term $1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=1}^{L-1}\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)$ cannot be always 0 for all samples in a mini-batch. This means that even if the weight is very small, the gradient of the layer will always exist.

Since its introduction, residual learning has become a basic idea in the architecture design of most computer vision tasks. DnCNN [ZZC+17] is a pioneer in applying residual learning in image restoration. The architecture of DnCNN is shown in Figure 3.11. Instead of using many residual blocks as in ResNet, DnCNN utilizes a single residual unit to predict the residual image. Through this residual learning strategy, DnCNN can implicitly remove the latent clean image in the hidden layers to get an accurate residual image. Besides, the experiment in [ZZC+17] demonstrated that residual learning and batch normalization can benefit from each other, speeding up the training and boosting the denoising performance effectively at the same time.

3.3.3. Attention Mechanism

Attention plays a significant role in human perception because the human perception system can not process a whole scene at once. Inspired by this mechanism, much work has been developed to improve the performance of deep learning.

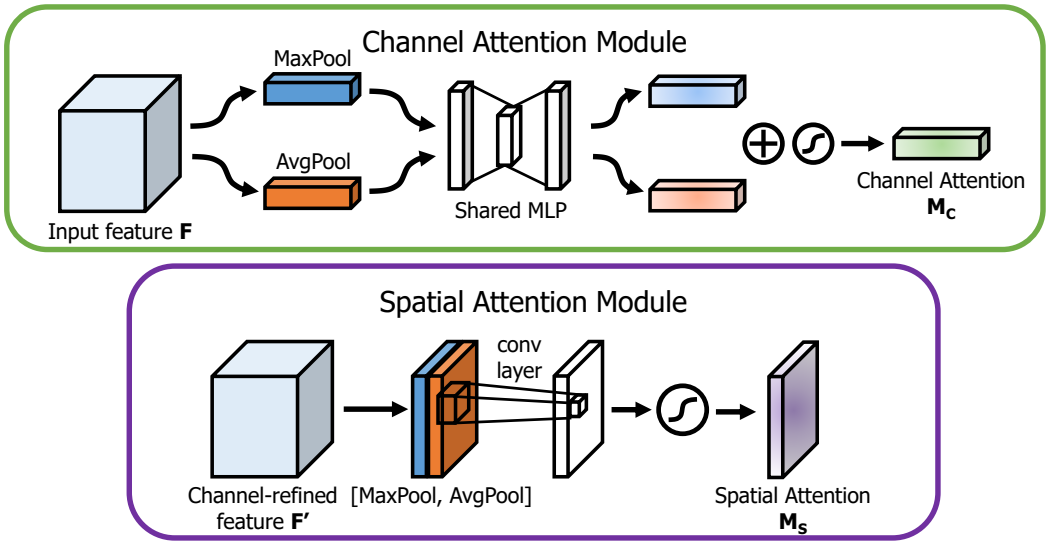


Figure 3.12.: **The structures of channel attention module and spatial attention module.** The channel attention module receives the outputs from max-pooling and average-pooling with a shared network, while the spatial attention module receives similar outputs which are pooled along the channel axis. Figure from [WPLK18].

In the computer vision community, the attention mechanism is often utilized by exploiting the inter-channel relationship (channel attention) or inter-spatial relationship (spatial attention) of image features [WPLK18]. The structures of the channel attention module and spatial attention module are illustrated in Figure 3.12. For the channel attention, max pooling and average pooling are applied to compute the channel attention maps efficiently. When using both poolings, parameters are also saved by using shared multi-layer perceptrons. Such a shared representation can be effective because both aggregated channel features lie in the same semantic embedding space. Next, the attention maps are multiplied to the input feature map for adaptive feature refinement. The spatial attention module focuses on localizing the informative part of the image features and is complementary to channel attention. The spatial attention scores are computed by applying average pooling and max pooling along the channel axis, followed by concatenating into an efficient feature descriptor. Then this descriptor applied by one convolution and sigmoid function to get the final attention map.

Attention mechanism can also be utilized in various image restoration tasks. The information in the low-quality images often has abundant low-frequency and valuable high-frequency components. While the low-frequency components are always kept during restoration, the high-frequency components would usually contain both annoying noise and the edge and texture details. Therefore, utilizing the attention mechanism to extract informative information for image restoration would be a promising solution.

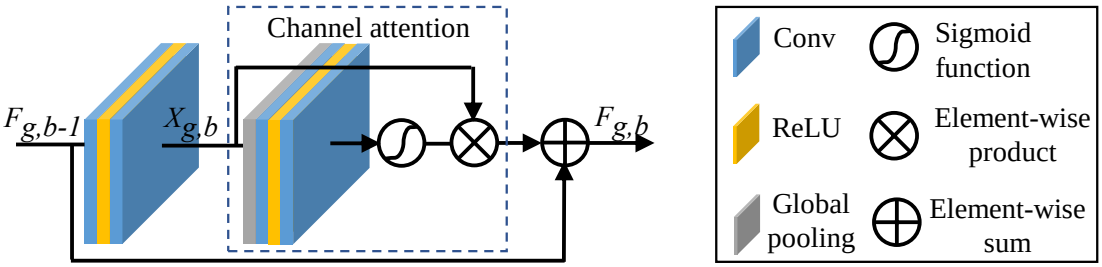


Figure 3.13.: **Residual channel attention block.** Figure from [ZLL+18].

In [ZLL+18], a residual channel attention block (RCAB) is proposed for image super-resolution. As illustrated in Figure 3.13, the channel attention is integrated with residual blocks. For the b -th residual block in g -th residual groups, the procedure is written by

$$F_{g,b} = F_{g,b-1} + R_{g,b}\left(X_{g,b}\right) \cdot X_{g,b} \tag{3.20}$$

where $F_{g,b}$ and $F_{g,b-1}$ are the input and output of RCAB. $R_{g,b}$ is the function of channel attention. The residual component $X_{g,b}$ is given by

$$X_{g,b} = W_{g,b}^2 \delta\left(W_{g,b}^1 F_{g,b-1}\right) \tag{3.21}$$

where $W_{g,b}^1$ and $W_{g,b}^2$ are weights of two convolutional layers in RCAB.

Inspired by the ideas utilizing attention mechanism in image restoration, in our work, we proposed a quality factor-aware residual attention block, whose attention scores are given by the quality factor of JPEG images, to flexibly guide the image restoration procedure.

3.3.4. Generative Adversarial Network

Generative adversarial networks (GANs) [GPAM+14] are a machine learning approach which generates new data with the same data distribution as the training set in a unsupervised way. The generative network usually learns a mapping from a latent space to a data distribution and generates new data as candidates while the discriminative network distinguishes if the canditates generated form generative network are true data distribution.

The training goal of the generative networks is to fool the discriminator by producing novel candidates so that the discriminative network will think they have true data distribution. The training procedure is illustrated in Figure 5.9. More formally, the networks are trained jointly in a so-called minimax game with the following objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{3.22}$$

where G and D denote the generator and discriminator respectively. The variable $\boldsymbol{z}$ is the randomly sampled noise.

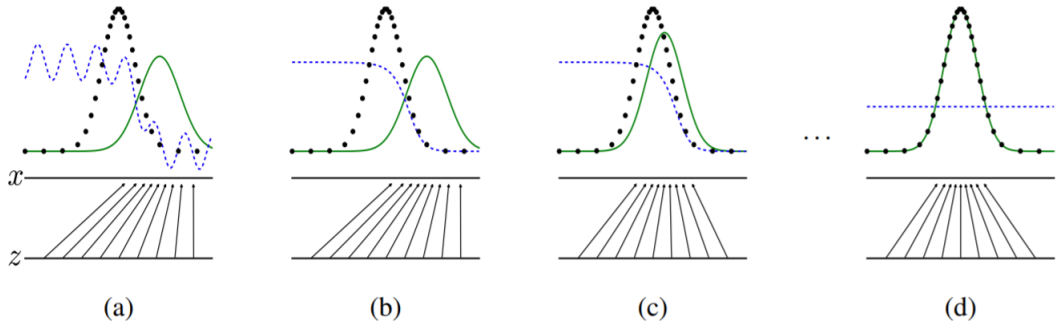


Figure 3.14.: **The process of training GAN.** From left to right: generative distribution p_{model} approaches the data distribution p_{data} and discriminator (blue dashed line) tries to distinguish between them (low values if x is fake and high values otherwise). Upon convergence (given G and D have sufficient capacity), the generator can't further improve as $p_{\text{model}} = p_{\text{data}}$ and the discriminator is unable to distinguish between the two distributions, i.e., $D(x) = \frac{1}{2}$. Figure from [GPAM+14].

A problem for training the generator is that the gradient is dominated by a region where the samples are already good and very flat when the sample is likely to be fake. However, particularly the bad fake images should be used to train and improve the generator. The solution is that instead of minimizing the likelihood of the discriminator to be correct, the likelihood of the discriminator to be wrong is maximized.

Upon convergence, $p_{\text{model}}(\mathbf{x})$ learned by the generator approximates the real data distribution $p_{\text{data}}(\mathbf{x})$. After training, we can use the generator network to generate new images by feeding random noise $\boldsymbol{z}$ through it.

In our thesis, we use the generative adversarial network to help add realistic texture details to the restored images.

4

Proposed Method

In this chapter, we introduce our novel flexible blind artifacts removal network, namely FBAR, and present its advantage over other state-of-the-art methods, especially for real complex JPEG images. In Section 4.1, we will present the architecture of our proposed FBAR. By analysing the limitation of existing blind methods on double JPEG compression, we address the importance of flexibility in Section 4.2. To make a full blind model, we propose a dominant quality estimation algorithm, which can correct the quality factor in double JPEG compression, in Section 4.3. We compare our method with other design choices in Section 4.4.

4.1. Flexible Blind Artifacts Removal Network

The overall architecture of our proposed method is illustrated in Figure 4.1. FBAR is an end-to-end model which takes a JPEG compressed image as input and directly generates the output image. Specifically, FBAR comprises four components: decoupler, QF predictor, flexible controller, and image reconstructor. The network is fairly straightforward, with each component designed to achieve a specific task.

Decoupler: The decoupler aims to extract deep features and decouple latent quality factor from the input image. The decoupler involves four scales, each of which has an identity skip connection to the reconstructor. Residual blocks are adopted in each scale, and each residual block is composed of two 3×3 convolution layers with ReLU activation in the middle. 2×2 strided convolutions are adopted for the downscaling operations. The number of output channels in each layer from the first to the third scale is set to 64, 128, 256, respectively. We set the number of channels in the fourth scale to 576, which is then split into 512 channels and 64 channels for image reconstruction and quality factor prediction branches. The image branch processes the encoded deep image features with 512 channels. Unlike the traditional design of U-Net-based networks with the same

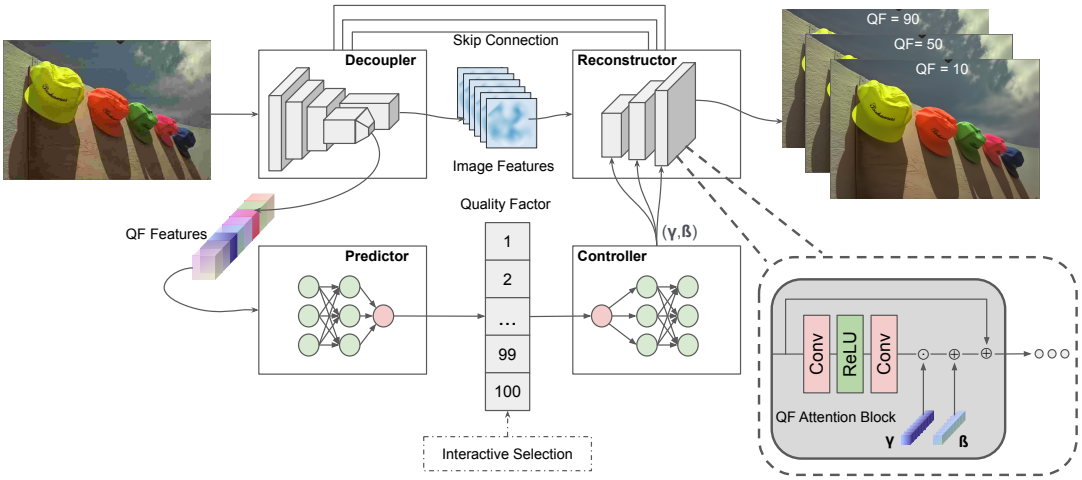


Figure 4.1.: **The architecture of the proposed FBAR for JPEG artifacts removal.** FBAR consists of four parts, i.e., decoupler, quality factor predictor, flexible controller and image reconstructor. The decoupler extracts the deep features from the input corrupted JPEG image and then splits them into image features and QF features which are subsequently fed into the reconstructor and predictor, respectively. The controller gets the estimated QF from the predictor and then generates QF embeddings. The QF attention block enables the controller to make the reconstructor produce different results according to different QF embeddings. More importantly, the predicted quality factor can be changed with interactive selections to have a balance between artifacts removal and details preservation, according to user preference.

number of blocks in each scale, increasing the blocks of bottleneck layers brings more gains on performance with less running time than other scales. We thus empirically set the number of residual blocks from the first scale to the fourth scales as 2, 2, 4, 8 to keep a balance between performance and inference speed. The output of the image branch is fed into the reconstructor. The quality factor branches first use residual blocks to extract higher-level information. Because the quality factor is a single value for each image, we adopt a global average pooling layer to get the global information from the image features.

Quality Factor Predictor: The QF predictor is a 5-layer multilayer perceptron (MLP) that takes as input the 64-dimensional QF features and produces an estimated quality factor QF_{est} of the compressed image. We set the number of nodes in each hidden layer as 512 for a better prediction. For small patch size during training, it is likely that the patch only includes the limited information and corresponds to multiple quality factors so that the quality factor can not be

accurately estimated, which may lead to an unstable training process. Therefore, we use the L1 loss function to avoid too much penalty for such outliers. Let N be the batch size during training and i the ith sample of each batch, the loss for quality factor estimation in each batch can be written as:

$$\mathcal{L}_{\mathrm{QF}} = \frac{1}{N}\sum_{i=1}^{N}\left\| QF_{\mathrm{est}}^{i} - QF_{\mathrm{gt}}^{i} \right\|_1 . \tag{4.1}$$

Flexible Controller: The flexible controller is a 3-layer MLP and takes as input the quality factor, representing the degree of compression of the targeted image. The controller aims to learn an embedding of the given quality factor that can be fused into the reconstructor for flexible control. Inspired by recent research in spatial feature transform [PLWZ19, WYDL18], the controller learns a mapping function that outputs a modulation parameter pair $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ which embeds the given quality factor. Specifically, the first two layers of MLP generate shared intermediate conditions, which is then split into three parts corresponding to the three scales in the reconstructor. In the last layer of MLP, we learn different parameter pairs for different scales in reconstructor whereas shared $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ are broadcasted to the QF attention block within the same scale.

Image Reconstructor: The image reconstructor includes three scales and consists of QF Attention Blocks. Image reconstructor receives image features from decoupler and quality factor embedding parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ to generate the restored clean image. The QF attention block is an important component of the reconstructor. The number of QF attention block from the smallest scale is 4, 2, 2. The learned parameter pair $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ adaptively influences the outputs by applying an affine transformation spatially to each intermediate feature maps inside the QF attention block of each scale.

After obtaining $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ from the controller, the transformation is carried out by scaling and shifting feature maps of a specific layer:

$$\boldsymbol{F}_{\mathrm{out}} = \boldsymbol{\gamma} \odot \boldsymbol{F}_{\mathrm{in}} \oplus \boldsymbol{\beta}, \tag{4.2}$$

where $\boldsymbol{F}_{\mathrm{in}}$ and $\boldsymbol{F}_{\mathrm{out}}$ denote the feature maps before and after the affine transformation, and $\odot$ is referred to element-wise multiplication, i.e., Hadamard product.

Given N training samples within a batch, the goal of the image reconstructor is to minimize the following L1 loss function between reconstructed image $\mathbf{I}_{\mathrm{rec}}$ and the original ground-truth image $\mathbf{I}_{\mathrm{gt}}$:

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{N}\sum_{i=1}^{N}\left\| \mathbf{I}_{\mathrm{rec}}^{i} - \mathbf{I}_{\mathrm{gt}}^{i} \right\|_1 . \tag{4.3}$$

Overall, the complete training objective can be written as:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{rec}} \cdot \mathcal{L}_{\text{rec}} + \lambda_{\text{QF}} \cdot \mathcal{L}_{\text{QF}}, \tag{4.4}$$

where λ_{rec} and λ_{QF} are the scaling factors, which control the balance between image reconstruction and quality factor estimation.

4.2. Flexibility for Real-World JPEG Artifatcs

In this section, we will analyse the difference between single and double JPEG compression and show how the subsequent image manipulation further disrupts a single compressed image.

Let us observe the appearances of JPEG images with different compression settings in Figure 4.2. We set the shift as (4, 4) in non-aligned cases. For QF = (70, 10), (10, 70) and (70, 10), the blocking effects are similar to single compression with QF = 10: the edges of 8 × 8 blocks are apparent. However, in the case of (10, 70), much information is lost, but the clear blocking edges no longer exist.

We test the restored results of these images using representative blind methods DnCNN [ZZC+17] and QGAC [ELDS20]. As shown in Figure 4.2, in cases of QF = 70, 10, (70, 10), the blocking effects are largely removed. When QF = (10, 70), QGAC failed to work because this method extracts the quantization table from the JPEG image, but JPEG images only have the most recent quantization table information. Therefore, QGAC regards the quality factor of this image as 70, so the reconstructed image almost keeps unchanged. DnCNN uses only pixel information, so it does not have this limitation. However, in this case of non-aligned double JPEG compression when QF1 = 10 and QF2 = 70, we can see that both methods almost fail to remove the artifacts.

Since our FBAR is also a pixel-based blind method like DnCNN but can predict the latent quality factor, it can be used to explain the behavior behind a blind method. We test the failed image using our FBAR. Not surprisingly, we get a similar, almost unchanged reconstructed result, but we find the predicted quality factor is 70. We continue to test other images with non-aligned double JPEG compression and QF1 < QF2, finding that the predicted quality factor is always close to QF2. This is to say, blind methods trained with single JPEG compression image pairs are always misled by the appearance of non-aligned double JPEG images with QF1 < QF2. Our experimental part will show that we can solve this problem by correcting the quality factor to QF1, demonstrating the need for a flexible network.

In summary, we classify double JPEG compression into two categories: simple and complex compression. Simple compression corresponds to non-aligned double JPEG with QF1 > QF2 and all aligned double JPEG compression, which is actually equivalent to single JPEG compression from the perspective of restoration. Complex compression corresponds to non-aligned double JPEG with QF1

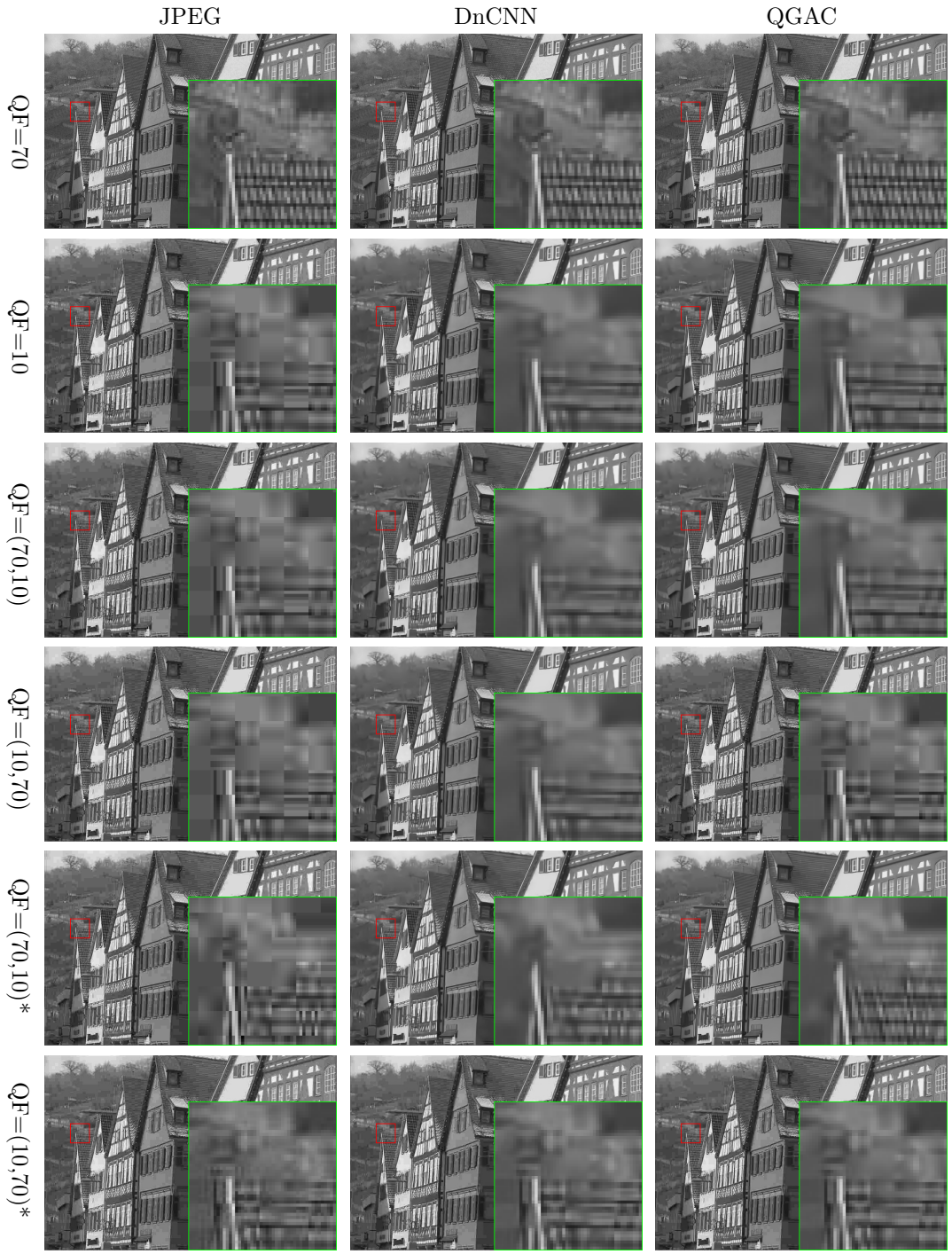


Figure 4.2.: **Visual comparison of an example of JPEG images with different degradation settings and their restored results by DnCNN and QGAC.** QF = (QF1, QF2) denotes that the image is firstly compressed with QF1 and then compressed with QF2. '*' means that there exists a shift between blocks of two compressions.

$\leq$ QF2, meaning that composite artifacts occur in this circumstance. Besides, we test images with these degradation settings by a recent double JPEG compression algorithm [PCAL18], finding that only images with non-aligned double JPEG with QF1 $\leq$ QF2 is identified, which further support our views.

4.3. Dominant Quality Factor Estimation

From the previous analysis, we have known that it is crucial to infer the smallest quality factor during the degradation history. Although many quality factor estimation methods exist, they are based on the DCT domain and need prior information from JPEG format. For general practical application, we seek to directly solve this problem from pixel-domain, independent of the image format.

By utilizing the property of JPEG compression, we find that if an image $\mathbf{I}$ is compressed to $\mathbf{I}'$ by QF1. Then we can apply another JPEG compression by QF2 to $\mathbf{I}'$ to get $\mathbf{I}''$. If the mean square error of $\mathbf{I}'$ and $\mathbf{I}''$ is zero, then QF2 = QF1. Therefore, for the single compressed JPEG image, we can easily get the correct quality factor by searching for the quality factor with the global minimum MSE from all possible candidates between 1 and 99, as shown in the left diagram of Figure 4.3. We further extend this method to challenging non-aligned cases when QF1 $<$ QF2, where our predictor trained with single JPEG compression always predicts the quality factor as QF2, although QF1 brings the dominant degradation. Let us apply the third JPEG, which is aligned with the first one.

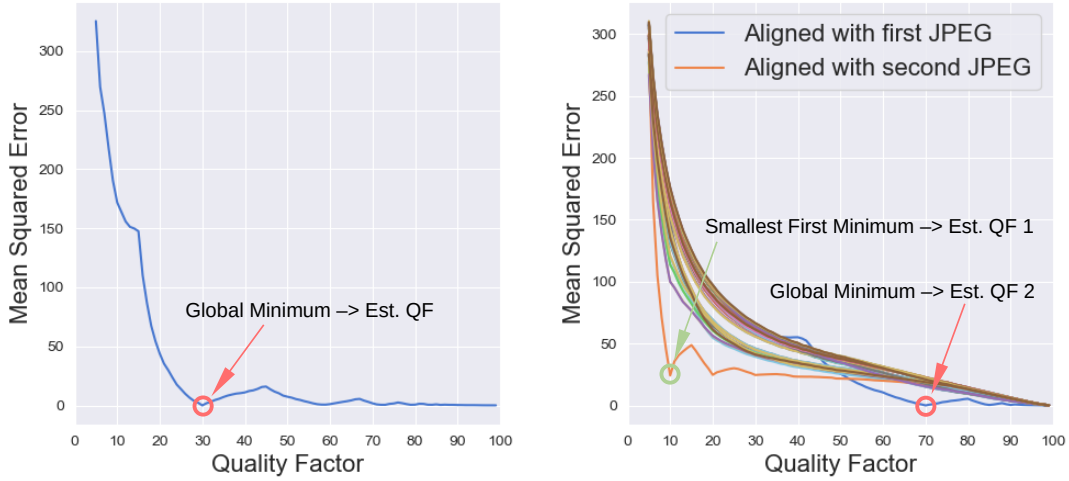


Figure 4.3.: **The curve of PSNR between the targeted JPEG image and the image after applying another JPEG compression over different quality factors.** Left: The targeted image is compressed only once, with QF = 30. Right: The targeted image is degraded by QF1 = 10, shift = (4, 4), QF2 = 70.

We can find that generally, the MSE curve has a minimum at QF1. Since the shift value is unknown, we search in the range of 0 to 7 in the two dimensions of an image, respectively.

The right diagram of Figure 4.3 shows an example with QF1 = 10, QF2 = 70 and shift = (4, 4), where all the 64 MSE curves are plotted. It can be seen that the smallest first minimum of all curves corresponds to the QF1 and location aligned with the first JPEG whereas the global minimum lies at the curve aligned with the second JPEG and corresponds to the QF2. We require that the MSE of the smallest first minimum be smaller than a threshold T to have more robust results. We empirically set T to 30 in our experiment.

4.4. Comparison with Other Design Choices

In the following, we will clarify the differences between the proposed FBAR and two alternative design choices.

FBAR vs. A Blind Model without QF Predictor: As we have discussed in Section 4.2, although the pure blind model performs favorably for single JPEG artifacts removal without knowing the quality factor, it does not generalize well to real corrupted images whose artifacts are much more complex. FBAR can be viewed as multiple deblockers and can control the trade-off between JPEG artifacts removal and details preservation. Besides, our FBAR with quality factor prediction has moderately better performance for artifacts removal than the pure blind one with about 0.05dB gain on average for the Classic5 dataset [ZEP10], possibly because the predicted quality factor provides additional information to the reconstruction network. We have similar observations in image super-resolution [RSRB15] and denoising [ZZZ18a].

FBAR vs. Cascaded QF Prediction and Non-blind Deblocking Model: It is also possible to design a QF predictor cascaded by a non-blind method. However, our method enjoys some benefits compared with such cascaded design: First, for accurate quality estimation, a convolutional network starting from the same scale as the input image is needed, which would increase the total model size and cost more training and inference time. Instead, we only add a relative small prediction branch. Second, our decoupler shared parameters for QF estimation and image reconstruction, which accelerates the convergence of predicting QF. On the contrary, in cascaded design, inaccurate QF estimation would lead to an unstable training process. It might be a solution to train a QF predictor and then freeze it to train the second part for reconstruction. Nevertheless, it would cost more training time than our joint training schedule. Third, in cascaded networks, the predicted parameter is treated as the input of the second part and propagates through the whole encoder-decoder architecture. Instead, our predicted parameter QF is the only input to the decoder part. We can change the QF to adjust

different outputs during inference without the need to change the encoded image features, which saves more than half of the inference time.

4.5. Further Improvements

Our approach can be extended for further improvements. We try to design a more realistic degradation model to augment the training data to deal with real images, and use the techniques based on GANs to improve the perceptual quality of images.

4.5.1. Design a Realistic Degradation Model

Existing methods assume the corrupted images are compressed only once, which is given by Equation 3.12. However, in real world, most images on the Internet are compressed many times. The misalignment of data distributions between training and real images will reduce the performance in the real application.

To tackle the problem of double JPEG compression, we can also augment our training data using images with double JPEG compression. For double JPEG compression, the degradation model is given by:

$$\mathbf{y} = \text{JPEG}(\text{shift}(\text{JPEG}(\mathbf{x}, \text{QF1})), \text{QF2}) \tag{4.5}$$

For shift operation, we randomly remove the first h rows and w columns of the image after first compression, and $0 \leq h, w \leq 7$, QF1 $\leq$ QF2. Therefore, images degraded with this model are always with double JPEG compression artifacts.

Now, the training data includes both single and double JPEG images. Please note that it is not necessary to include the case of QF1 $>$ QF2 or increase the number of images with aligned double JPEG compression, because they are actually equivalent to single JPEG compression, as we have shown in Section 4.2.

When trained with double JPEG compressed images, we set the weight of quality factor loss to zero. Then the dominant quality factor can be trained in an unsupervised way. We name the FBAR model with augmented training data as FBAR-A.

4.5.2. Improve Perceptual Quality by GAN

Since restored images tend to give blurry results especially when the quality factor is small. Following the steps of ESRGAN [WYW+18], we fine-tune our model with additional VGG perceptual loss [JAFF16] and relativistic GAN loss [JM18] to improve the perceptual quality. The fine-tuned model is denoted as FBAR-GAN.

The difference between a relativistic GAN and a standard GAN is that a relativistic discriminator was proposed to estimate the probability that the real data

is more realistic than a fake data. The loss function of the discriminator is given by:

$$L_D^{Ra} = -\mathbb{E}_{x_r}[\log(D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f}[\log(1 - D_{Ra}(x_f, x_r))] \tag{4.6}$$

In relativistic GAN, the generator benefits from the gradients from both generated data and real data in adversarial training, and its loss function is given by:

$$L_G^{Ra} = -\mathbb{E}_{x_r}[\log(1 - D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f}[\log(D_{Ra}(x_f, x_r))] \tag{4.7}$$

However, in original GAN, only generated part takes effect.

The perceptual loss is the distance between intermediate feature representations from a pre-trained deep neural networks for classification. Here the feature is from VGG [SZ15] network pretrained on ImageNet [KSH12].

The overall loss function of our generator is written as:

$$L_G = \alpha L_{\text{percep}} + \beta L_G^{Ra} + \gamma L_{\text{rec}} + \eta L_{QF} \tag{4.8}$$

where $\alpha, \beta, \gamma, \eta$ are weights of different loss terms.

5 Experiments

In this chapter, we will focus on experimentally evaluating our proposed methods, compared with state-of-the-art approaches, for JPEG artifacts removal. First, we introduce the training and testing datasets in Section 5.1 and standard evaluation metrics in Section 5.2. Then we present the implementation and training details Section 5.3. Next, we show the qualitative and quantitative results on synthetic datasets by single, double, and triple JPEG compression respectively and provide new state-of-the-art results in Section 5.4. Furthermore, in Section 5.5, we compare our proposed method with other methods on real JPEG images to demonstrate the effectiveness of our solution for real application. We also do ablation studies in Section 5.6 and compare the running time in Section 5.7. Finally, we present the results of our model fine-tuned with GANs in Section 5.8 and examples of practical applications in Section 5.9.

5.1. Datasets

In this section, we introduce the training and testing datasets employed in our experiments. First, we show the widely used datasets for training and testing in image restoration tasks. We will also present our proposed Meme dataset, which is specially collected for comparisons on real-world JPEG images.

5.1.1. Existing Datasets

In our experiments, we train our methods on training sets of DIV2K [AT17] and Flickr2K [TAVG+17], and test the results on Classic5 [ZEP10], LIVE1 [She05], and BSDS500 [MFTM01]. The descriptions of these datasets are given as follows.

Training Data

DIV2K: DIV2K dataset contains 1000 2K resolution images, with 800 for training, 100 for validation, and 100 for testing, respectively. These images were collected from the Internet, with special care to the image quality, the diversity of sources including sites and cameras, the image contents and the copyrights. These images are of high quality both aesthetically and in terms of few corruptions. Besides, the image contents cover a large diversity ranging from natural scenarios, cities, villages to people, animals and plants.

Flickr2K: Flickr2K dataset contains 2650 high-resolution images collected from flickr.com using Flickr API. This dataset also has a large range of diversity including human, fauna, flora, landscape.

Testing Data

Classic5: Classic 5 dataset contains five classic grayscale images (Lena, Barbara, Boat, Man, Couple) commonly used for image quality assessment tasks. The resolution of these five images is all 512×512.

LIVE1: LIVE1 is a publicly released dataset that contains 29 images for image quality assessment. Different from Classic5, these 29 images are in color and with different resolutions.

BSDS500: BSDS500 dataset is a publicly released database for image segmentation that contains 200 training images, 100 validation images, and 200 test images. These images are also in color and with different resolutions. Despite the original purpose for image segmentation, this dataset has also been widely used for evaluating the task of JPEG artifacts removal.

It should be pointed out that some methods adopted the 100 validation images for testing while some ways adopted the 200 test images for testing. For a fair comparison, this difference needs to be noted and stated clearly. In our experiments, we use the 200 test images for evaluation.

5.1.2. Proposed Meme Dataset

The ultimate goal in JPEG artifacts removal is to restore the real JPEG images with unknown degradation history. However, most existing methods only evaluate their methods on synthetic datasets, which may have highly different data distribution from real-world JPEG images. In Section Section 4.2, we have shown that in double JPEG compression with QF1 $<$ QF2, there exist complex compression artifacts, where current blind methods fail to work. Although we can still generate synthetic datasets, including double or more JPEG compression, real JPEG images can still have more complex degradation such as resizing, spatially variant artifacts, and other types of noise. Besides, different degrada-

Figure 5.1.: **Samples of our proposed Meme dataset**. This dataset contains 400 JPEG images, which have various compression artifacts.

tion sequences and combinations make it difficult to set up an effective synthetic dataset. Therefore, it is necessary to propose a new dataset consisting of real JPEG images.

As JPEG is the most widely used image format on the Internet, it is natural to think about directly downloading JPEG images from the Internet randomly. However, as we have discussed in Section 4.2, if the most recent quality factor is the smallest one, this image is equivalent to a single JPEG compressed image. As a result, such kind of single JPEG compressed image occupies the majority on the Internet, making it a class imbalance problem. However, the minor multiple compressed images can be significant or of high value for some users, so the collected images should contain as many complex artifacts as possible. Meme images are a good example that satisfies our requirements. Most meme images are usually generated by adding words to an image, which is easy to generate double compression artifacts. Some meme images are a combination of multiple images, which may lead to spatially variant artifacts. Besides, most meme images often contain either cartoon characters or human faces. While original clean images of the human face always have much high-frequency texture information such as skin and hair, clean cartoon images usually have uniform textures, without much high-frequency information. Therefore, the artifacts on cartoon images and blurred information on the human face would be very easy to observe. Since the main challenges in JPEG image restoration is how to remove artifacts and preserve necessary texture details at the same time, we can use cartoon images to exam the performance of artifacts removal and use human face images to exam the ability to preserve textures, making meme images pretty suitable for evaluating real JPEG image restoration.

We propose a Meme dataset by collecting 400 meme images from the Internet. To satisfy the diversity, we get these images by simply typing the keyword 'meme' on the Google Images search engine and save them if they are JPEG format.

It should be pointed out that due to the lack of ground truth, it will be impossible to evaluate using common reference-based evaluation metrics such as PSNR and SSIM. Using a non-reference image quality indicator is a possible solution but can be inaccurate. Through our evaluation, we find that obvious visual difference can be observed between different methods, making the visual comparison a more reliable evaluation method for our proposed dataset.

5.2. Evaluation Metrics

In this section, we introduce the evaluation metrics used in our experiments. For synthetic datasets, PSNR, SSIM [WBSS04] and PSNR-B [YB10] are calculated following the conventions of the research community in JPEG artifacts removal. PSNR and SSIM are commonly used in various image restoration tasks, where PSNR-B is specially designed for JPEG deblocking tasks. For real JPEG images, non-reference quality indicators NIQE and BEF are employed, as there are not ground truth images.

PSNR: The peak signal-to-noise ratio (PSNR) is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the quality of its representation. It is the most popular evaluation measure to make a comparison between images or signals. Since most signals have a very wide dynamic range, PSNR is usually written as a logarithmic quantity.

In image restoration, given a $H \times W$ groundtruth monochrome image $\mathbf{I}_{\text{gt}}$ and reconstructed result $\mathbf{I}_{\text{rec}}$, we firstly define the mean squared error (MSE) between them, which is written as:

$$\text{MSE} = \frac{1}{HW} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} [\mathbf{I}_{\text{rec}}(i,j) - \mathbf{I}_{\text{gt}}(i,j)]^2 \tag{5.1}$$

Next, the PSNR (in dB) is defined as:

$$\begin{aligned} \text{PSNR} &= 10 \cdot \log_{10} \left(\frac{\text{MAX}_\text{I}^2}{\text{MSE}} \right) \\ &= 20 \cdot \log_{10} \left(\frac{\text{MAX}_\text{I}}{\sqrt{\text{MSE}}} \right) \\ &= 20 \cdot \log_{10} (\text{MAX}_\text{I}) - 10 \cdot \log_{10} (\text{MSE}) \end{aligned} \tag{5.2}$$

where $\text{MAX}_\text{I} = 2^B - 1$ is the maximum possible pixel value within an image and B is the length of bits per pixel. In our experiments, we use 8-bit images, so the value of MAX_I is 255, and typical values for a 8-bit image is between 30 and 50 dB, where higher PSNR means better image quality.

SSIM: The structural similarity index measure (SSIM) is also a widely used

image quality measure. It is a full-reference image quality assessment (FR-IQA), which means the measurement is based on an original clean image as a reference.

SSIM is usually calculated using a sliding Gaussian window of size 11×11 or a block window of size 8×8. The SSIM between two windows x and y with the same size $N \times N$ is:

$$\mathrm{SSIM}(x, y) = \left[l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x, y)^{\gamma} \right] \tag{5.3}$$

This formula is based on the three components, i.e. luminance (l), contrast (c) and structure (s). Each component is calculated by:

$$\begin{aligned} l(x, y) &= \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \\ c(x, y) &= \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \\ s(x, y) &= \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \end{aligned} \tag{5.4}$$

where: μ and σ^2 are the average and variance of the window. σ_{xy} is the covariance of x and y. $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are used to avoid the denominator closet to zero. L is the length of bits per pixel. By default, we set $k_1 = 0.01, k_2 = 0.03$.

To simplify, α, β, γ can be set to 1, and $c_3 = c_2/2$, then Equation 5.3 can be written as:

$$\mathrm{SSIM}(x, y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{5.5}$$

Finally, the mean SSIM bewteen the ground truth $\mathbf{I}_{\mathrm{gt}}$ as the reference and the reconstructed output $\mathbf{I}_{\mathrm{rec}}$ as the distorted image over all M local windows is calculated by:

$$\mathrm{MSSIM}(\mathbf{I}_{\mathrm{gt}}, \mathbf{I}_{\mathrm{rec}}) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{SSIM}\left(\mathbf{x}_j, \mathbf{y}_j\right) \tag{5.6}$$

The difference between SSIM with PSNR is that PSNR only estimates the absolute error without considering the structural information within an image. Actually, spatially closed pixels should have stronger inter-dependencies than distant pixels. These inter-dependencies contains significant information about the image structure.

BEF and PSNR-B: The blocking effect factor (BEF) and peak signal-to-noise ratio including blocking effect factor (PSNR-B) were proposed in [YB10] to measure the deblocked JPEG images.

First, the mean boundary pixel squared difference D_B and the mean nonboundary pixel squared difference D_B^C of the image $\mathbf{y}$ are defined as:

$$D_B(\mathbf{y}) = \frac{\sum_{(y_i, y_j) \in \mathcal{H}_B} \left(y_i - y_j\right)^2 + \sum_{(y_i, y_j) \in \mathcal{V}_B} \left(y_i - y_j\right)^2}{N_{H_B} + N_{V_B}} \tag{5.7}$$

$$D_B^C(\mathbf{y}) = \frac{\sum_{(y_i,y_j)\in\mathcal{H}_B^C}\left(y_i - y_j\right)^2 + \sum_{(y_i,y_j)\in\mathcal{V}_B^C}\left(y_i - y_j\right)^2}{N_{H_B^C} + N_{V_B^C}} \tag{5.8}$$

where:

- N_H and N_V are the horizontal and vertical dimensions of the $N_H \times N_V$ image $\mathbf{y}$.
- $\mathcal{H}$ and $\mathcal{V}$ are the set of horizontal and vertical neighboring pixel pairs in image $\mathbf{y}$.
- $\mathcal{H}_B \subset \mathcal{H}$ is the set of horizontal neighboring pixel pairs that lie across a block boundary and $\mathcal{H}_B^C$ is the set of horizontal neighboring pixel pairs not lying across a block boundary.
- $\mathcal{V}_B \subset \mathcal{V}$ is the set of vertical neighboring pixel pairs that lie across a block boundary and $\mathcal{V}_B^C$ is the set of vertical neighboring pixel pairs not lying across a block boundary.

Then the blocking effect factor is defined as:

$$\mathrm{BEF}(\mathbf{y}) = \eta \cdot \left[D_B(\mathbf{y}) - D_B^C(\mathbf{y})\right] \tag{5.9}$$

where

$$\eta = \begin{cases} \frac{\log_2 B}{\log_2\left(\min(N_H,N_V)\right)}, & \text{if } D_B(\mathbf{y}) > D_B^C(\mathbf{y}) \\ 0, & \text{otherwise} \end{cases} \tag{5.10}$$

By this definition we can find that BEF as a function of block size, which is based on that the visibility of blocking effects increases with the block size.

The mean squared error including blocking effects (MSE-B) for reference groundtruth image $\mathbf{I}_{\mathrm{gt}}$ and reconstructed image $\mathbf{I}_{\mathrm{rec}}$ is then defined as:

$$\text{MSE-B}(\mathbf{I}_{\mathrm{gt}}, \mathbf{I}_{\mathrm{rec}}) = \mathrm{MSE}(\mathbf{I}_{\mathrm{gt}}, \mathbf{I}_{\mathrm{rec}}) + \mathrm{BEF}(\mathbf{I}_{\mathrm{rec}}) \tag{5.11}$$

Finally, the PSNR-B is defined as:

$$\text{PSNR-B}(\mathbf{I}_{\mathrm{gt}}, \mathbf{I}_{\mathrm{rec}}) = 10\log_{10}\frac{\mathrm{MAX}_{\mathrm{I}}^2}{\text{MSE-B}(\mathbf{I}_{\mathrm{gt}}, \mathbf{I}_{\mathrm{rec}})} \tag{5.12}$$

According to the above definitions, BEF itself can be used as a non-reference image quality indicator. However, it is usually efficient for measuring the blockiness, but not for the image quality. For PSNR, it measures the image quality but does not consider the blocking effects. Therefore, the combination of PSNR and BEF can not only assess image quality but also measure the blocking effects.

NIQE: The natural image quality evaluator (NIQE) is a completely blind image quality indicator that does not need the reference image. NIQE measures the distance between the natural scene statistic based features extracted from the input image and the features obtained from an image used to train the model. The features are modelled as multidimensional Gaussian distributions. A smaller score of NIQE means a better image quality.

5.3. Implementation and Training Details

For fair experimental comparison, all training and evaluation processes are conducted on the Y' channel of $Y'C_BC_R$ space and the JPEG compressed images are generated by MATLAB JPEG encoder, according to the common settings of data generation. Following [ELDS20], we use DIV2K [AT17] and Flickr2K [TAVG+17] as training data. We randomly crop one million patch pairs with size 96×96, and the compressed images are with random quality factor from 10 to 90. To optimize the parameters of FBAR, we adopt the Adam solver [KB15] with mini-batch size 64. The learning rate starts from 1×10^{-4} and decays by a factor of 0.5 every 4×10^4 iterations and finally ends with 1.25×10^{-5}. Upon convergence, we finetune our model with training patches with the size 256×256 for a few epochs. We train our model with PyTorch on two NVIDIA GeForce GTX 2080Ti GPUs. It takes about two days to obtain the FBAR model.

5.4. Experiments on Synthetic JPEG Images

5.4.1. Single JPEG Compression

We first evaluate the performance of the proposed FBAR on the images with single JPEG compression. We test on the commonly used benchmarks: Classic5 [ZEP10], LIVE1 [She05] and the test set of BSDS500 [MFTM01]. We compare our proposed FBAR with ARCNN [DDCLT15], MWCNN [LZZ+18], DnCNN [ZZC+17], DCSC [FZW+19], QGAC [ELDS20]. It should be pointed out that ARCNN, MWCNN train a single network for each specific value of quality factor and DCSC is trained with quality factors from 10 to 40. Only DnCNN, QGAC and our FBAR cover a full range of quality factors.

We calculate the PSNR, SSIM, and PSNR-B for quantitative assessment. The quantitative results are shown in Table 5.1. Our method has moderately better results with MWCNN, which trains each model for a specific quality factor and better results than other blind methods.

For subjective comparisons, some restored images of different approaches on the LIVE1 dataset have been presented. As can be seen in Figure 5.2, the results of our FBAR are more visually pleasing.

We also trained our model on RGB channels, referred to as FBAR-C. We compare FBAR-C with QGAC, which is a state-of-the-art method especially for color JPEG image restoration. The evaluation is made on LIVE1 dataset and testset of BSDS500. Although QGAC is specially designed for color JPEG image artifacts removal, we still get better performance. The result is shown in Figure 5.2.

Table 5.1.: **PSNR|SSIM|PSNRB results of ARCNN*, MWCNN*, DnCNN, DCSC, QGAC, and our proposed FBAR on single JPEG compression.** Please note that the methods marked with '*' train a specific model for each quality factor. The best two results are highlighted in red and blue colors, respectively.

Dataset	Quality Factor	JPEG	ARCNN*	MWCNN*	DCSC	DnCNN	QGAC	FBAR (Ours)
Classic5	10	27.82|0.760|25.21	29.03 |0.793|28.76	30.01|0.820|29.59	29.62|0.810|29.30	29.40|0.803|29.13	29.84|0.812|29.43	30.06|0.821|29.77
	20	30.12|0.834|27.50	31.15|0.852|30.59	32.16|0.870|31.52	31.81|0.864|31.34	31.63|0.861|31.19	31.98|0.869|31.37	32.23|0.871|31.71
	30	31.48|0.867|28.94	32.51|0.881|31.98	33.43|0.893|32.62	33.06|0.888|32.49	32.91|0.886|32.38	33.22|0.892|32.42	33.46|0.894|32.75
	40	32.43|0.885|29.92	33.32|0.895|32.79	34.27|0.906|33.35	33.87|0.902|33.30	33.77|0.900|33.23	34.05|0.905|33.12	34.27|0.906|33.43
	50	33.20|0.898|30.76	—	—	—	34.46|0.911|33.94	34.73|0.915|33.74	34.94|0.916|34.04
	60	33.96|0.910|31.57	—	—	—	35.11|0.920|34.53	35.38|0.923|34.25	35.57|0.924|34.51
	70	34.98|0.923|32.71	—	—	—	35.97|0.930|35.35	36.24|0.933|35.00	36.42|0.934|35.23
	80	36.44|0.938|34.43	—	—	—	37.09|0.942|36.41	37.48|0.946|36.07	37.64|0.946|36.26
	90	39.37|0.963|38.07	—	—	—	38.54|0.951|37.85	40.13|0.968|38.49	40.21|0.967|38.64
LIVE1	10	27.77|0.773|25.33	28.96|0.808|28.68	29.69|0.825|29.32	29.34|0.818|29.01	29.19|0.812|28.90	29.51|0.825|29.13	29.70|0.826|29.43
	20	30.07|0.851|27.57	31.29|0.873|30.76	32.04|0.889|31.51	31.70|0.883|31.18	31.59|0.880|31.07	31.83|0.888|31.25	32.06|0.889|31.61
	30	31.41|0.885|28.92	32.67|0.904|32.14	33.45|0.915|32.80	33.07|0.911|32.43	32.98|0.909|32.34	33.20|0.914|32.47	33.45|0.915|32.87
	40	32.35|0.904|29.96	33.61|0.920|33.11	34.45|0.930|33.78	34.02|0.926|33.36	33.96|0.925|33.28	34.16|0.929|33.36	34.42|0.930|33.79
	50	33.16|0.918|30.86	—	—	—	34.77|0.934|34.06	34.95|0.939|34.09	35.23|0.940|34.54
	60	33.98|0.929|31.79	—	—	—	35.57|0.945|34.85	35.76|0.948|34.82	36.04|0.949|35.30
	70	35.13|0.943|33.14	—	—	—	36.67|0.955|35.96	36.86|0.958|35.87	37.13|0.958|36.38
	80	36.87|0.958|35.26	—	—	—	38.29|0.967|37.62	38.48|0.969|37.43	38.74|0.969|37.99
	90	40.39|0.977|39.41	—	—	—	41.38|0.981|40.66	41.60|0.982|40.48	41.82|0.982|41.20
BSDS500	10	27.80|0.768|25.10	29.10|0.804|28.73	29.61|0.820|29.14	29.32|0.813|28.91	29.21|0.809|28.80	29.46|0.821|28.97	29.61|0.821|29.25
	20	30.05|0.849|27.22	31.28|0.870|30.55	31.92|0.885|31.15	31.63|0.880|30.92	31.53|0.878|30.79	31.73|0.884|30.93	31.92|0.885|31.27
	30	31.37|0.884|28.53	32.67|0.902|31.94	33.30|0.912|32.34	32.99|0.908|32.08	32.90|0.907|31.97	33.07|0.912|32.04	33.28|0.912|32.42
	40	32.30|0.903|29.49	33.55|0.918|32.78	34.27|0.928|33.19	33.92|0.924|32.92	33.85|0.923|32.80	34.01|0.927|32.81	34.24|0.928|33.22
	50	33.10|0.917|30.38	—	—	—	34.67|0.935|33.60	34.82|0.938|33.53	35.05|0.936|33.97
	60	33.92|0.929|31.28	—	—	—	35.47|0.944|34.35	35.62|0.947|34.21	35.85|0.947|34.67
	70	35.08|0.943|32.46	—	—	—	36.52|0.955|35.25	36.67|0.957|35.01	36.89|0.957|35.54
	80	36.85|0.959|34.12	—	—	—	38.00|0.966|36.46	38.13|0.968|36.08	38.34|0.968|36.70
	90	40.21|0.978|36.83	—	—	—	40.62|0.980|38.39	40.78|0.981|37.77	40.91|0.981|38.62

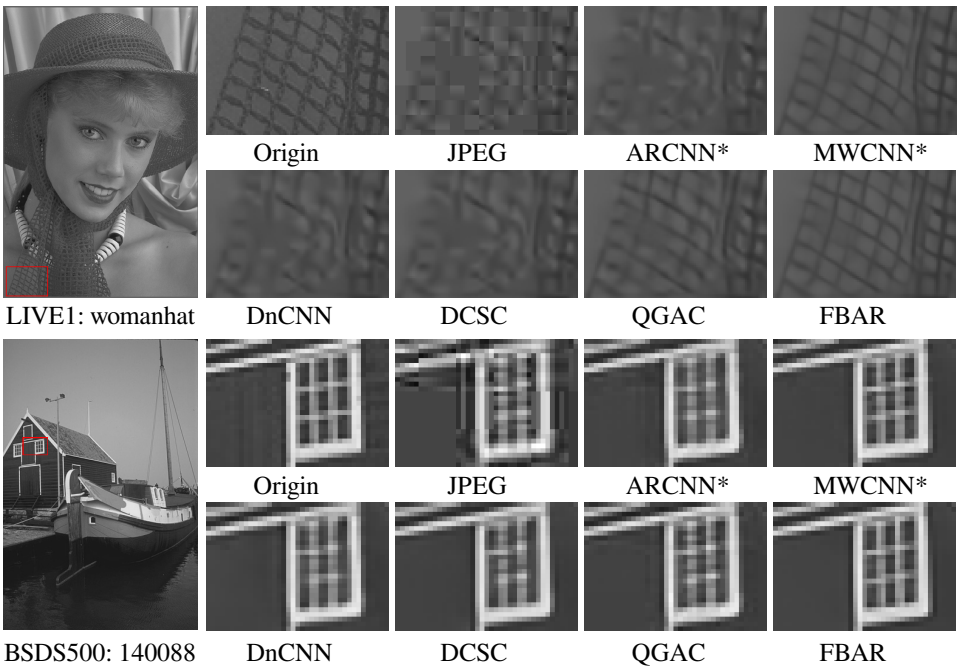


Figure 5.2.: **Visual results on single JPEG compression.**

Table 5.2.: **PSNR|SSIM|PSNRB results of QGAC and FBAR-C on color JPEG images with single compression.** Although QGAC is also designed for color JPEG image restoration, our method trained with the same condition still performs better on PSNR by 0.2 dB on average.

Dataset	Quality Factor	JPEG	QGAC	FBAR (Ours)
LIVE1	10	25.69|0.743|24.20	27.62|0.804|27.43	27.77|0.803|27.51
	20	28.06|0.826|26.49	29.88|0.868|29.56	30.11|0.869|29.69
	30	29.37|0.861|27.84	31.17|0.896|30.77	31.42|0.898|30.92
	40	30.28|0.882|28.84	32.05|0.912|31.61	32.32|0.913|31.79
	50	31.03|0.897|29.67	32.77|0.923|32.29	33.04|0.924|32.48
	60	31.77|0.909|30.51	33.47|0.932|32.96	33.74|0.933|33.15
	70	32.77|0.924|31.69	34.40|0.943|33.86	34.67|0.944|34.08
	80	34.23|0.941|33.45	35.68|0.955|35.12	35.96|0.956|35.38
	90	36.86|0.963|36.45	37.64|0.970|37.10	38.07|0.970|37.46
BSDS500	10	25.84|0.741|24.13	27.74|0.802|27.47	27.86|0.800|27.52
	20	28.21|0.827|26.37	30.01|0.869|29.53	30.18|0.869|29.57
	30	29.57|0.865|27.72	31.33|0.898|30.70	31.50|0.898|30.75
	40	30.52|0.887|28.69	32.25|0.915|31.50	32.42|0.915|31.57
	50	31.31|0.902|29.57	33.03|0.927|32.22	33.18|0.927|32.28
	60	32.11|0.916|30.46	33.82|0.937|32.92	33.95|0.937|32.99
	70	33.21|0.931|31.72	34.87|0.949|33.88	34.98|0.949|33.96
	80	34.80|0.948|33.72	36.41|0.962|35.36	36.47|0.962|35.43
	90	37.62|0.970|37.14	39.12|0.980|38.02	39.15|0.980|38.04
Average		31.29|0.885|29.93	32.96|0.913|32.35	33.16|0.914|32.48

5.4.2. Double JPEG Compression

The focus of our thesis is to remove the real complex JPEG artifacts, where double JPEG compression is one important step towards this goal. So it is necessary also to evaluate the performance of current state-of-the-arts and our proposed methods on images with double JPEG compression. We compare our methods with blind methods: DnCNN, DCSC, QGAC.

The comparison is conducted using different combinations of quality factors (QF1, QF2) on Classic5 dataset. Each original image is JPEG compressed with a quality factor QF1, decompressed, cropped by a random shift (i, j), with $0 \leq i \leq 7, 0 \leq j \leq 7$ with respect to the upper left corner, and JPEG compressed with another quality factor QF2. We generate 10 double compressed JPEG image for each original one.

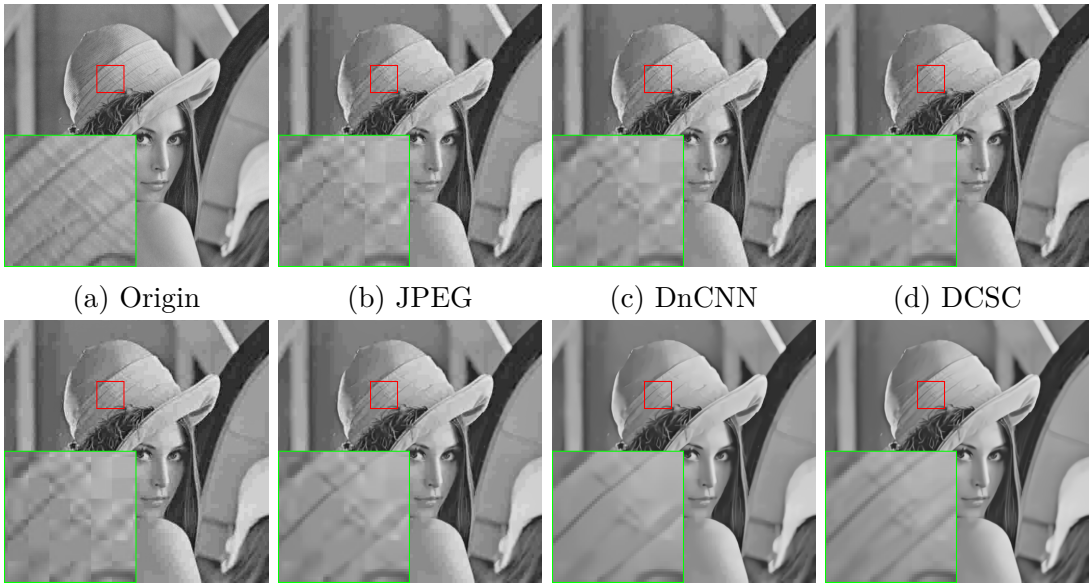
The numerical and visual results are reported in Table 5.3 and Figure 5.3. As shown in Table 5.3, the order of first quality factor QF1 and QF2 significantly impacts other methods and our FBAR. When changing the order of QF1 and QF2, although the differences of PSNR values are generally smaller than 0.2 dB, a significant drop in performance can be seen on other methods and our FBAR. However, our proposed FBAR can help our analysis and explain the behavior of blind methods. We find that the predicted quality factor is always close to the most recent quality factor for non-aligned double JPEG compressed images. Because our FBAR and other methods are all trained with single compressed

Table 5.3.: **PSNR|SSIM|PSNRB comparisons of DnCNN, DCSC, QGAC, our proposed FBAR, FBAR-D and FBAR-A on double JPEG compression.** The best two results are highlighted in red and blue colors. It can be found that DnCNN, QGAC and FBAR, which were trained with single JPEG images with a wide range of quality factors from 10 to 90 do not have a good performance when Q1 < Q2. DCSC trained with QFs from 10 to 40 performs relatively better as expected, according to our previous analysis in the main paper. By correcting the predicted quality factor, FBAR-D boosts the performance in the type of Q1 < Q2. FBAR-A further improves these results by augmenting the training data.

Type	Quality Factor	JPEG	DnCNN	DCSC	QGAC	FBAR (Ours)	FBAR-D (Ours)	FBAR-A (Ours)
QF1>QF2	(30,10)	27.43|0.748|24.96	29.09|0.796|28.83	29.30|0.803|29.02	29.48|0.812|29.08	29.72|0.815|29.42	29.72|0.815|29.42	29.69|0.813|29.42
	(50,10)	27.74|0.756|25.18	29.40|0.802|29.15	29.62|0.809|29.33	29.83|0.819|29.45	30.07|0.821|29.79	30.07|0.821|29.79	29.98|0.819|29.74
	(70,10)	27.74|0.757|25.15	29.37|0.802|29.10	29.60|0.810|29.30	29.82|0.819|29.42	30.05|0.822|29.75	30.05|0.822|29.75	29.96|0.819|29.69
	(90,10)	27.83|0.758|25.22	29.43|0.802|29.17	29.63|0.808|29.34	29.86|0.818|29.48	30.08|0.820|29.80	30.08|0.820|29.80	29.98|0.817|29.73
	(50,30)	30.67|0.848|28.51	32.25|0.875|31.91	32.40|0.877|32.02	32.44|0.880|31.83	32.70|0.882|32.19	32.70|0.882|32.19	32.76|0.882|32.40
	(70,30)	31.07|0.859|28.68	32.66|0.884|32.21	32.82|0.886|32.34	32.93|0.889|32.18	33.20|0.891|32.55	33.20|0.891|32.55	33.15|0.890|32.65
	(90,30)	31.39|0.867|28.89	32.89|0.888|32.48	33.05|0.890|32.60	33.20|0.894|32.51	33.45|0.896|32.86	33.45|0.896|32.86	33.33|0.893|32.89
	(70,50)	32.56|0.885|30.42	34.08|0.904|33.60	34.20|0.906|33.75	34.24|0.908|33.35	34.50|0.909|33.69	34.41|0.907|33.75	34.55|0.909|33.94
	(90,50)	32.99|0.896|30.54	34.36|0.911|33.80	34.41|0.911|33.87	34.61|0.915|33.59	34.84|0.916|33.89	34.62|0.913|33.85	34.74|0.914|34.01
	(90,70)	34.63|0.917|32.54	35.81|0.928|35.28	35.62|0.926|35.26	36.02|0.931|34.84	36.24|0.932|35.10	36.03|0.929|35.16	36.17|0.931|35.33
QF1=QF2	(10,10)	26.63|0.708|24.67	28.08|0.761|27.89	28.16|0.765|27.97	28.26|0.772|28.00	28.44|0.775|28.26	28.44|0.775|28.26	28.66|0.780|28.51
	(30,30)	30.06|0.831|28.20	31.52|0.860|31.20	31.64|0.862|31.32	31.61|0.864|31.09	31.85|0.866|31.43	31.85|0.866|31.43	32.08|0.869|31.78
	(50,50)	31.95|0.871|30.15	33.39|0.892|33.04	33.55|0.894|33.20	33.43|0.895|32.75	33.67|0.897|33.08	33.73|0.896|33.26	33.88|0.898|33.46
	(70,70)	33.33|0.900|31.80	34.67|0.916|34.26	34.75|0.917|34.40	34.69|0.917|33.88	34.95|0.919|34.17	34.98|0.918|34.31	35.13|0.920|34.54
	(90,90)	37.33|0.946|36.43	37.88|0.948|37.34	37.20|0.946|36.82	38.40|0.956|37.32	38.66|0.956|37.53	38.63|0.956|37.51	38.80|0.957|37.89
QF1<QF2	(10,30)	27.60|0.748|26.52	28.63|0.782|28.48	28.61|0.781|28.44	28.40|0.778|28.10	28.52|0.781|28.33	29.02|0.791|28.90	29.51|0.806|29.40
	(10,50)	27.70|0.754|27.10	28.52|0.781|28.38	28.61|0.782|28.47	28.25|0.775|27.96	28.38|0.778|28.20	29.19|0.795|29.06	29.64|0.811|29.54
	(10,70)	27.81|0.757|27.31	28.57|0.781|28.40	28.78|0.787|28.64	28.18|0.772|27.82	28.46|0.780|28.31	29.39|0.800|29.30	29.87|0.817|29.76
	(10,90)	27.80|0.757|27.63	28.47|0.778|28.41	28.89|0.791|28.87	27.98|0.766|27.81	29.04|0.797|29.00	29.69|0.810|29.66	29.95|0.820|29.89
	(30,50)	30.84|0.850|29.42	32.18|0.874|31.96	32.37|0.877|32.13	32.06|0.874|31.57	32.31|0.877|31.89	32.61|0.880|32.30	32.83|0.883|32.60
	(30,70)	30.98|0.856|30.16	32.12|0.875|31.97	32.42|0.880|32.28	31.80|0.873|31.45	32.09|0.876|31.85	32.69|0.882|32.49	32.88|0.886|32.71
	(30,90)	31.27|0.862|30.90	32.28|0.877|32.12	32.80|0.885|32.65	31.61|0.870|31.28	32.59|0.884|32.42	33.16|0.890|33.00	33.20|0.891|33.02
	(50,70)	32.32|0.883|31.19	33.58|0.900|33.32	33.80|0.903|33.58	33.43|0.900|32.91	33.68|0.902|33.20	34.03|0.905|33.61	34.15|0.906|33.85
	(50,90)	32.89|0.892|32.43	33.96|0.905|33.77	34.31|0.909|34.10	33.33|0.900|32.90	34.06|0.907|33.76	34.60|0.912|34.31	34.63|0.912|34.33
	(70,90)	34.57|0.917|34.13	35.62|0.926|35.43	35.64|0.926|35.44	35.17|0.924|34.80	35.68|0.928|35.25	36.10|0.930|35.73	36.13|0.931|35.81
Average		30.69|0.833|29.13	31.95|0.858|31.66	32.09|0.861|31.81	31.96|0.861|31.41	32.29|0.865|31.83	32.50|0.868|32.09	32.63|0.871|32.28

images, our quality factor prediction can reveal how general blind methods work under the unseen scenario. To further support our views, let us look at the results of DCSC. Since DCSC is trained with quality factor from 10 to 40, it would assume that all the input images are with quality factor from 10 to 40. Therefore, it performs generally better than DnCNN, QGAC and FBAR when QF1 < QF2. Despite some benefits for double JPEG compression, it should be pointed out that it is not reasonable to use a model trained with low quality factors to tackle all kinds of JPEG images. When dealing with relative high-quality images, it tends to give more blurry results.

To show the effectiveness of our proposed dominant quality factor estimation algorithm, we set the input controller as the value got from this algorithm. We refer to this combination as FBAR-D. Table 5.3 shows that by correcting the predicted quality factor, we largely improve the PSNR in QF1 < QF2 by around 0.5 dB. Despite some failed examples in (70, 50), the PSNR is already relative high (more than 34 dB) in this case.



(a) Origin (b) JPEG (c) DnCNN (d) DCSC

(e) QGAC (f) FBAR (g) FBAR-D (h) FBAR-A

Figure 5.3.: **Qualitative comparisons of an example 'Lena' from Classic5 dataset with synthetic non-aligned double JPEG compression.** In this image, QF1 = 10, QF2 = 70 and the shift = (7, 4). It can be seen that clear blocking effects remain after restoration by other blind methods and our FBAR. We got a significantly better visual result by correcting the predicted QF from our dominant QF prediction algorithm. By augmenting the training data, FBAR-A can keep more texture detail.

To demonstrate the flexibility of FBAR, we set the quality factor manually by traversing from 10 to 90. The test image is firstly compressed by quality factor 30, followed by a shift of (4,4) and then a second JPEG compression with quality factor 70. We plot the curve between the PSNR and quality factor in Figure 5.4. We can find that the result provided by FBAR-D is very close to the theoretical best result. We also show the visual results with quality 10, 30 and 90 in Figure 5.5. We can observe that when the quality factor is 10, artifacts around the words are clearly removed, but the bricks become blurred. When quality factor is 90, the bricks keep much texture information. However, the artifacs around the words are not removed. By setting different quality factors, users can get results with different perception qualities and make an interactive selection according to their preference.

We also investigate if our FBAR can predict the dominant quality factor by augmenting the training data. To achieve this, we augment the training data with double JPEG compression with QF1 $<$ QF2 upon the convergence of training with single JPEG compression. We set the weight of quality factor loss function $\lambda_{QF} =$

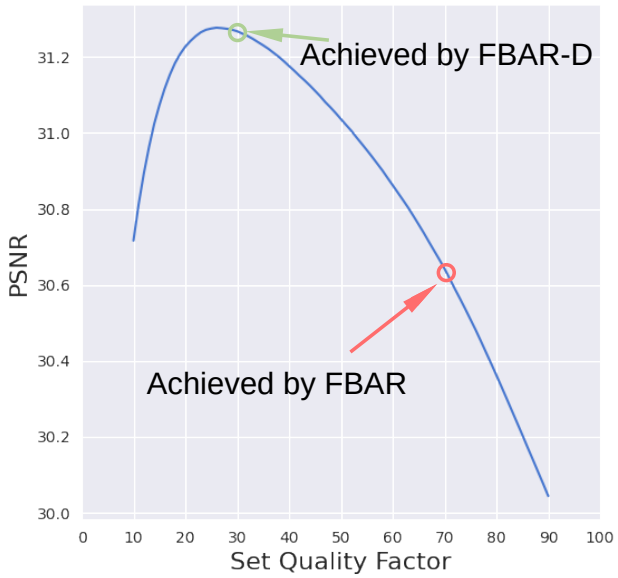


Figure 5.4.: **PSNR results over the set quality factor from 10 to 90.**

(a) JPEG (b) QF = 10 (c) QF = 30 (d) QF = 90

Figure 5.5.: **An example to show the flexibility of FBAR on the image.** The image 'LIVE1: cemetry' is double compressed by QF1 = 30 and QF2 = 70 and shift = (4, 4). Visual comparisons of setting the controllable quality factor as 10, 30 and 90. Although the artifacts around the words can be effectively removed when the QF is small, the texture on the bricks becomes blurred. Therefore, users can get the desired results through interactive selection by FBAR.

0 when training with double compressed JPEG images. After only around 800 iterations with batch size 64, we get our augmented version FBAR-A, predicting the dominant quality factor accurately in a self-supervised way. We also report the results of FBAR-A in Table 5.3 and Figure 5.3. We can find that FBAR-A further improves performance when QF1 $<$ QF2. The difficult case when QF1 = QF2 also sees an improvement by FBAR-A.

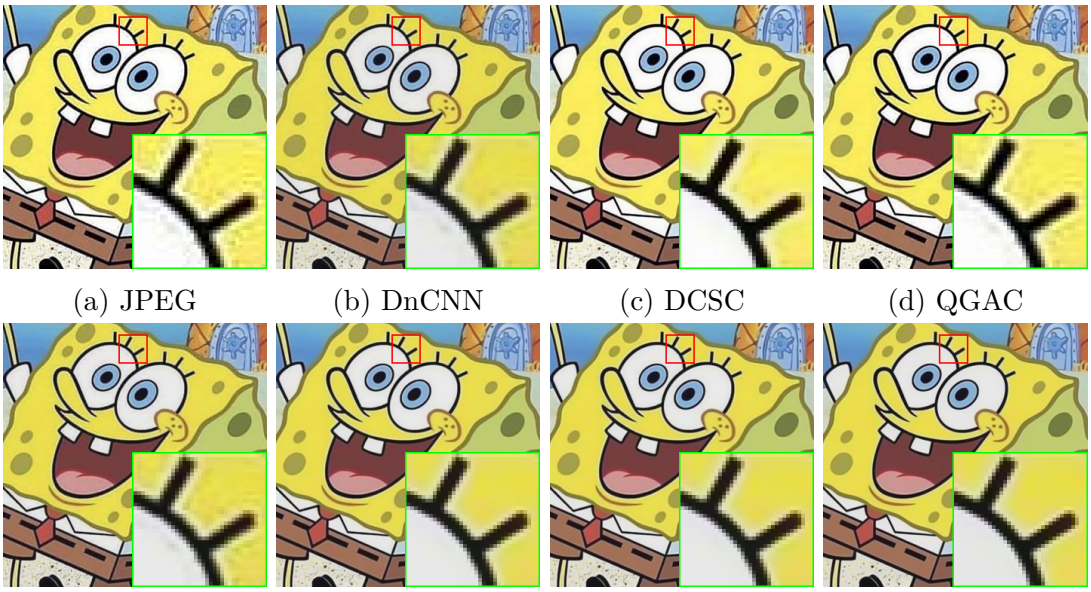
5.4.3. Triple JPEG Compression

To further show the performance of images with more complex degradation, we resize the generated synthetic double JPEG images from the Classic5 dataset. The scale was selected randomly from the range of (0.5, 2). Next, a third JPEG compression with quality factor randomly from 10 to 90 is applied to each image. Resizing between two JPEG compression is more complex than shifting because the previous 8×8 blocks would be destroyed by the resizing operation. To obtain the ground truth images, we apply the same resizing operation. The results are provided in Table 5.4. Please note that in this case, predicting the dominant quality factor becomes more difficult, so we include the results when the quality factor is set as the smallest one of triple compression, namely FBAR-D*, which can be obtained by interactive selection. Please note that FBAR, FBAR-D and FBAR-D* are the same model with different set quality factors, which demonstrates the flexibility of our proposed method.

Table 5.4.: **PSNR|SSIM|PSNRB comparisons of DnCNN, DCSC, QGAC, our proposed FBAR, FBAR-D and FBAR-D* on triple JPEG compression.** For each of the double JPEG compressed images, we apply a random resizing with a scale between 0.5 to 2, followed by third JPEG compression, with the quality factor randomly selected from 10 to 90. Since the dominant quality factor is difficult to predict in this case, we include the reachable approximate best result through interactive selection by setting the quality factor as the smallest one, namely FBAR-D*. The best two results are highlighted in red and blue colors.

Type	Quality Factor	JPEG	DnCNN	DCSC	QGAC	FBAR (Ours)	FBAR-D (Ours)	FBAR-D* (Ours)
QF1>QF2	(30,10)	29.27|0.802|28.27	30.07|0.829|29.84	30.13|0.830|29.91	29.89|0.825|29.52	29.99|0.827|29.68	30.21|0.832|29.95	30.61|0.842|30.43
	(50,10)	29.47|0.806|28.45	30.36|0.834|30.24	30.39|0.836|30.26	30.12|0.829|29.81	30.31|0.833|30.12	30.51|0.838|30.35	30.90|0.847|30.80
	(70,10)	29.58|0.811|28.57	30.43|0.838|30.28	30.49|0.840|30.34	30.28|0.836|30.01	30.36|0.837|30.15	30.71|0.846|30.54	31.00|0.851|30.88
	(90,10)	29.53|0.808|28.51	30.39|0.836|30.23	30.45|0.838|30.31	30.18|0.832|29.87	30.32|0.835|30.10	30.58|0.841|30.42	30.95|0.848|30.83
	(50,30)	32.27|0.875|30.73	33.40|0.897|33.13	33.56|0.901|33.31	33.43|0.900|32.96	33.61|0.902|33.19	33.81|0.905|33.45	33.85|0.906|33.50
	(70,30)	32.83|0.884|31.27	33.98|0.905|33.76	34.12|0.909|33.91	33.99|0.908|33.55	34.15|0.909|33.78	34.35|0.912|34.04	34.44|0.914|34.13
	(90,30)	32.76|0.888|31.06	33.94|0.909|33.68	34.14|0.913|33.90	34.04|0.913|33.62	34.25|0.915|33.90	34.39|0.916|34.09	34.44|0.917|34.13
	(70,50)	33.59|0.901|31.56	34.95|0.922|34.39	35.08|0.925|34.56	35.02|0.925|34.21	35.26|0.927|34.51	35.34|0.928|34.66	35.37|0.928|34.67
	(90,50)	33.92|0.903|31.77	35.23|0.924|34.86	35.35|0.927|35.04	35.38|0.928|34.80	35.57|0.929|35.06	35.66|0.930|35.24	35.66|0.930|35.19
	(90,70)	34.44|0.904|32.26	35.67|0.922|35.34	35.73|0.925|35.45	35.91|0.928|35.35	36.13|0.929|35.64	36.15|0.929|35.75	36.13|0.929|35.65
QF1=QF2	(10,10)	28.31|0.766|27.56	28.94|0.793|28.86	28.92|0.794|28.85	28.75|0.788|28.58	28.79|0.789|28.67	29.00|0.796|28.91	29.45|0.810|29.39
	(30,30)	31.91|0.867|30.60	32.95|0.890|32.75	33.06|0.893|32.86	32.85|0.890|32.50	33.04|0.892|32.76	33.18|0.894|32.91	33.31|0.897|33.06
	(50,50)	33.62|0.894|31.86	34.84|0.915|34.57	34.99|0.918|34.74	34.91|0.918|34.46	35.09|0.919|34.71	35.22|0.920|34.89	35.19|0.920|34.83
	(70,70)	34.44|0.908|32.36	35.71|0.927|35.41	35.89|0.931|35.62	35.91|0.931|35.41	36.11|0.933|35.69	36.26|0.934|35.94	36.12|0.933|35.70
	(90,90)	35.20|0.918|32.43	36.57|0.937|36.01	36.54|0.939|36.12	36.96|0.942|36.14	37.18|0.943|36.46	36.98|0.941|36.38	37.18|0.943|36.45
QF1<QF2	(10,30)	29.29|0.799|28.40	30.04|0.825|29.94	30.05|0.827|29.95	29.90|0.823|29.73	29.99|0.825|29.86	30.15|0.830|30.04	30.56|0.840|30.48
	(10,50)	29.29|0.804|28.40	30.13|0.831|30.01	30.18|0.833|30.06	29.98|0.829|29.76	30.08|0.831|29.90	30.42|0.840|30.28	30.69|0.844|30.57
	(10,70)	29.52|0.803|28.63	30.29|0.830|30.20	30.29|0.831|30.21	30.16|0.828|30.02	30.21|0.829|30.09	30.50|0.836|30.39	30.86|0.844|30.77
	(10,90)	29.65|0.813|28.84	30.39|0.837|30.28	30.43|0.838|30.31	30.22|0.834|30.01	30.28|0.835|30.11	30.56|0.842|30.42	30.96|0.850|30.85
	(30,50)	33.04|0.884|31.63	34.06|0.903|33.78	34.15|0.906|33.88	33.90|0.902|33.45	34.03|0.904|33.65	34.28|0.908|33.96	34.43|0.910|34.12
	(30,70)	32.74|0.881|31.31	33.75|0.900|33.48	33.88|0.904|33.63	33.74|0.902|33.31	33.89|0.904|33.51	33.98|0.905|33.63	34.20|0.909|33.85
	(30,90)	32.81|0.884|31.16	33.96|0.905|33.72	34.09|0.908|33.87	33.98|0.908|33.55	34.14|0.909|33.77	34.34|0.911|34.00	34.39|0.912|34.09
	(50,70)	32.91|0.886|30.91	34.25|0.910|33.91	34.39|0.913|34.08	34.40|0.914|33.86	34.61|0.916|34.14	34.69|0.917|34.30	34.68|0.917|34.23
	(50,90)	34.00|0.897|31.98	35.19|0.916|34.85	35.32|0.919|35.00	35.27|0.920|34.73	35.48|0.921|35.02	35.59|0.922|35.20	35.59|0.922|35.15
	(70,90)	34.54|0.911|32.10	35.89|0.930|35.51	36.00|0.933|35.68	36.16|0.935|35.57	36.39|0.936|35.89	36.32|0.935|35.90	36.40|0.936|35.90
Average		31.87|0.860|30.43	33.02|0.883|32.76	33.11|0.885|32.87	33.01|0.884|32.59	33.17|0.885|32.89	33.33|0.888|33.03	33.49|0.892|33.19

5.5. Experiments on Real-World JPEG Images



(a) JPEG (b) DnCNN (c) DCSC (d) QGAC

(e) FBAR (f) FBAR-D (g) FBAR-Q10 (h) FBAR-A

Figure 5.6.: **Qualitative results of an example from our Meme dataset.** It can be seen that our proposed FBAR-D and FBAR-A have the best visual quality.

Table 5.5.: **BEF and NIQE scores of different methods on Meme dataset.**

	JPEG	DnCNN	DCSC	QGAC	FBCNN	FBCNN-D	FBCNN-A	FBCNN-Q10
BEF	1.23	0.20	0.14	0.41	0.27	0.25	0.12	0.11
NIQE	13.26	14.01	13.83	14.09	14.05	14.07	13.88	13.88

Besides the above experiments on synthetic test images, we also conduct experiments on real images to demonstrate the effectiveness of the proposed FBAR. We collected 400 meme images from the Internet, which is introduced in Section 5.1.2. This kind of images is widely used in social media and often compressed many times. Since there are no ground-truth HR images, we only provide the visual comparison. Figure 5.6 shows a test example on our collected Meme dataset. More results can be seen in the appendix.

Since the ground truth images are not available, we evaluate the average scores of different methods over our Meme dataset using BEF [YB10] (Blocking Effect Factor) and NIQE [MSB12] (Naturalness Image Quality Evaluator) as non-reference image quality indicators in Table 5.5. Since these images are severely degraded, we also set the quality factor of FBAR as 10 (FBAR-Q10) to compare. Both indicators mean better quality when they are smaller. Please note

that both indicators can only work as a reference because they can not describe the image quality accurately and comprehensively. For example, NIQE prefers natural images, which are only part of the images in the real world. Besides, a blurry image would have a low BEF, despite the loss of many details. Therefore, non-reference image quality indicators designed for real JPEG images are needed to be proposed in future.

5.6. Ablation Studies

We do ablation studies on different factors which can influence the performance of our network, and the result is given. The experiments are conducted on our synthetic Classic5 dataset with (QF1, QF2) = (10, 90), and each original image gives 10 double compressed images with random shift values. We remove the quality factor predictor (FBAR w/o predictor), do not fine-tune with patches of the size 256×256 (FBAR w/o fine-tuning), set the number of residual blocks from the first to the fourth scale of decoupler all as 2 (FBAR-S1) and 4 (FBAR-S2). As a reminder, the number of blocks from the first to the fourth scale in our default settings is 2, 2, 4, 8. We also compare the result with FBAR with dominant quality factor prediction (FBAR-D) and with augmented training data (FBAR-A). Figure 5.6 demonstrates the effectiveness of our design choices. Especially when the dominant quality factor is successfully identified, the performance of non-aligned double JPEG compression can be largely improved, which provides an idea on how to deal with real complex corrupted images when prior knowledge about the degradation history is unknown.

Table 5.6.: **Ablation studies on different influencing factors.** The largest improvement occurs when our dominant quality factor prediction algorithm corrects the predicted quality factor.

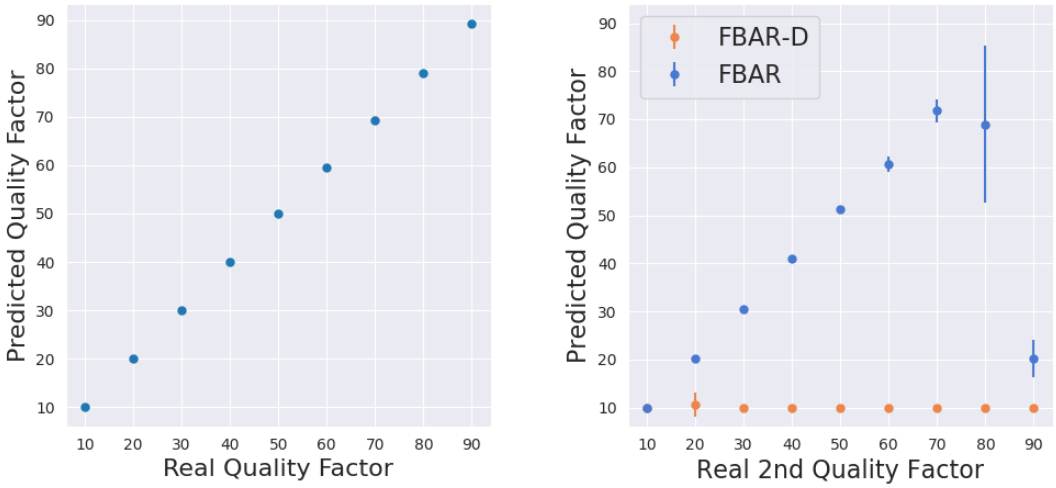
Experiments	PSNR	SSIM	PSNRB
FBAR w/o predictor	28.90	0.787	28.86
FBAR w/o fine-tuning	29.01	0.796	28.97
FBAR-S1	28.81	0.793	28.72
FBAR-S2	28.91	0.794	28.87
FBAR	29.04	0.797	29.00
FBAR-D	29.69	0.810	29.66
FBAR-A	29.95	0.820	29.89

We also show the results of quality factor prediction on the LIVE1 dataset with single and double JPEG compression in Figure 5.7. For single image compression, images are compressed with quality factors from 10 to 90 with a step of 10. For double JPEG compression, we fix the first quality factor QF1 = 10, and the second quality factor QF2 is set from 10 to 90 with a step of 10. To better show

the influence of non-aligned double JPEG compression, we set the shift value of double JPEG compression as (4, 4), as in this case, the 8×8 blocks from each compression are not overlapped to a relatively large degree.

As we can see in Figure 5.7, in the case of single compression, the predictor of FBAR can predict the latent quality factor accurately with almost negligible error. However, in a non-aligned double JPEG case, the predictor tends to give a result similar to QF2, although the main artifacts are introduced by the first JPEG compression. This observation explains why traditional blind methods trained with a full range of quality factors fail to work in non-aligned double JPEG compression with QF1<QF2: The network is misled by the additional artifacts introduced by QF2. We also find that when QF2 is large (i.e. QF2 = 90), the predicted quality factor is close to 20, which means the predictor can still recognize the main artifacts despite additional slight artifacts.

We also show the predicted quality factors given by our dominant quality factor prediction algorithm. It can be observed that although the network tends to give a result similar to QF2, the predicted quality factor from our algorithm is always close to the smaller QF1. By correcting the quality factor, FBAR-D can boost the performance in double JPEG restoration.



(a) Single JPEG (b) Double JPEG, QF1=10

Figure 5.7.: **Results of predicted quality factors on LIVE1. Left:** Images are single JPEG compressed. The predictor of our FBAR can give accurate results of the quality factors. **Right** Images are double JPEG compressed. The predicted quality factor from our network is always closet to QF2, excect for QF = 90. FBAR-D denotes FBAR with quality factor corrected by the dominant quality algorithm. We can see that our algorithm can always identify the smaller QF1.

5.7. Running Time Analysis

We show the runtime performance of our network compared to MWCNN [LZZ+18], DnCNN [ZZC+17], DCSC [FZW+19], QGAC [ELDS20], which we ran against, in Figure 5.8. Please note that we do not include ARCNN [DDCLT15] because its source code does not provide GPU acceleration. We measure the running time per frame (in seconds) on LIVE1 dataset on single NVIDIA 1080Ti GPU. Our FBAR achieved a good trade-off between performance increase and inference speed.

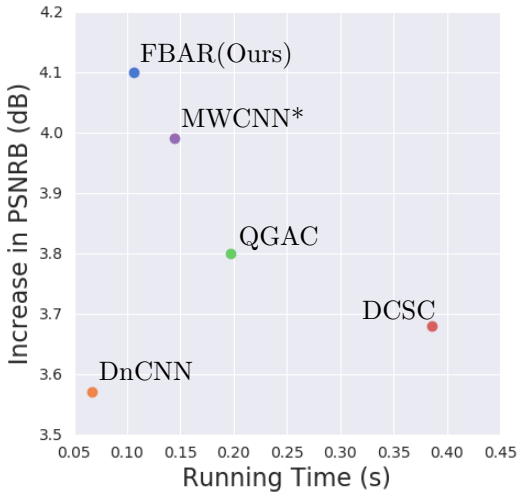


Figure 5.8.: **Increase in PSNR-B vs running time per frame.** It should be pointed out that MWCNN trains a separate model for a specific quality factor

5.8. Results of FBAR-GAN

Although our method has achieved favourable results regarding the provided numerical metrics, such as PSNR, PSNR measures the image quality only in a mathematical way and does not consider the human visual systems. For example, PSNR-oriented methods tend to give blurry results when the input image was compressed with a small quality factor. To increase the perceptual quality of our method, we use the techniques from GAN and perceptual loss to fine-tune our network trained for higher PSNR.

We show the results of our model fine-tuned with relativistic GAN loss and VGG perceptual loss, which can add realistic texture details. As we can see in Figure 5.9, our FBAR can remove the severe blocking effects effectively. However, the texture details on the fountain are lost, and the overall image looks

over-smoothed and blurry. On the contrary, our FBAR-GAN can remove the artefacts and contain realistic high-frequency texture details.

Please note that there is a difference in keeping texture details between our flexible model and with GAN. In our flexible model, we can change the quality factor to a larger value so that most high-frequency information from the original low-quality image remains. However, in FBAR-GAN, the texture information is generated by the generator to make the data distribution of the output closer to real images, so it is not from the input image.

(a) JPEG (b) FBAR (c) FBAR-GAN

Figure 5.9.: **Results of FBAR-GAN compared with JPEG and FBAR.** When the JPEG image is severely compressed, PSNR-oriented methods tend to give blurry result. Our FBAR-GAN can add realistic texture details to overcome this problem.

5.9. Applications

Since JPEG is the most widely used image format on the Internet, successful JPEG artifacts removal will benefit other computer vision tasks. This section will give examples of how our proposed method can improve the performance of single image super-resolution and object detection, which are representatives of low-level and high-level computer vision tasks, respectively.

5.9.1. Super-Resolution

USRNet [ZGT20] is a state-of-the-art method for single image super-resolution. However, as we can see from Figure 5.10, when the low-resolution image contains JPEG artifacts, the output would include many gridding effects, making the result worse than input. If the input image is processed by our FBAR, we can remove the JPEG artifacts. Then USRNet can provide a desirable high-resolution result.

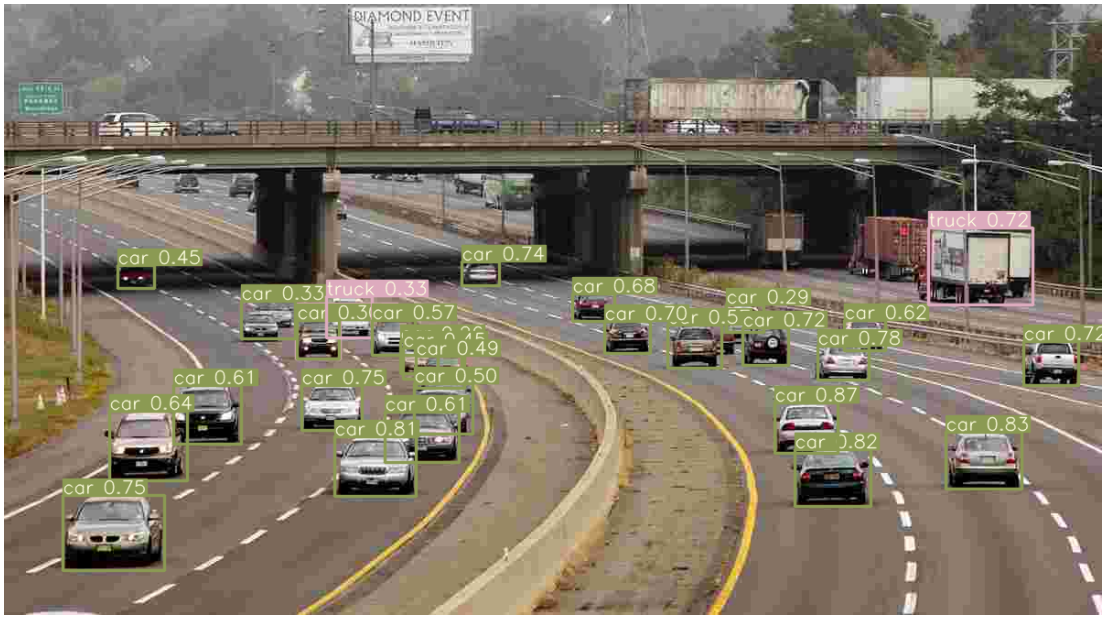
(a) JPEG (b) SISR of (a) by USRNet

(c) Artifacts Removal of (a) by FBAR (d) SISR of (c) by USRNet

Figure 5.10.: **Application of FBAR in image super-resolution.** Our JPEG artifacts removal method can help to improve the performance of single image super-resolution.

5.9.2. Object Detection

Most existing models for high-level vision tasks are trained using high quality images, which can lead to degraded performance when the input contains many JPEG artifacts. Our FBAR can also benefit object detection as a high-level vision task. Here we use the recent YOLOv5 as an example. By preprocessing the input JPEG images with our FBAR, we improve the performance of object detection. We improve the confidence scores, detect more objects. The misclassified car is also correctly identified.



(a) JPEG

(b) Ours

Figure 5.11.: **Application of FBAR in object detection.** Our JPEG artifacts removal method can help to improve the performance of object detection.

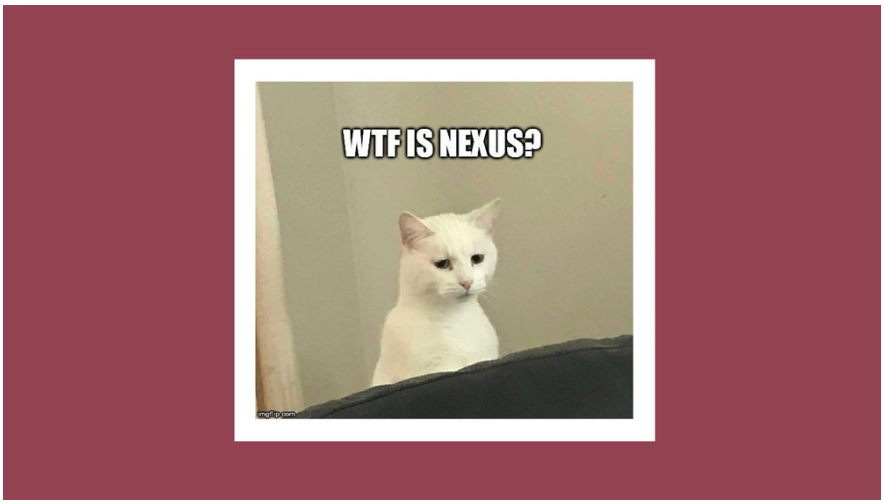
6 Conclusion

In this thesis, we proposed a flexible blind JPEG artifacts removal network (FBAR) for real JPEG image restoration. FBAR decouples the quality factor from the input image via a decoupler and then embeds the predicted quality factor into the subsequent reconstructor through a quality factor attention block for flexible control. The predicted quality factor can also be adjusted to achieve a balance between artifacts removal and details preservation. Besides, we address non-aligned double JPEG restoration tasks to take steps towards real JPEG images with severe degradations. Extensive experiments on single JPEG compressed images, the more general double compressed JPEG images and real-world JPEG images demonstrate the flexibility, effectiveness and generalizability of our proposed FBAR for restoring different kinds of degraded JPEG images.

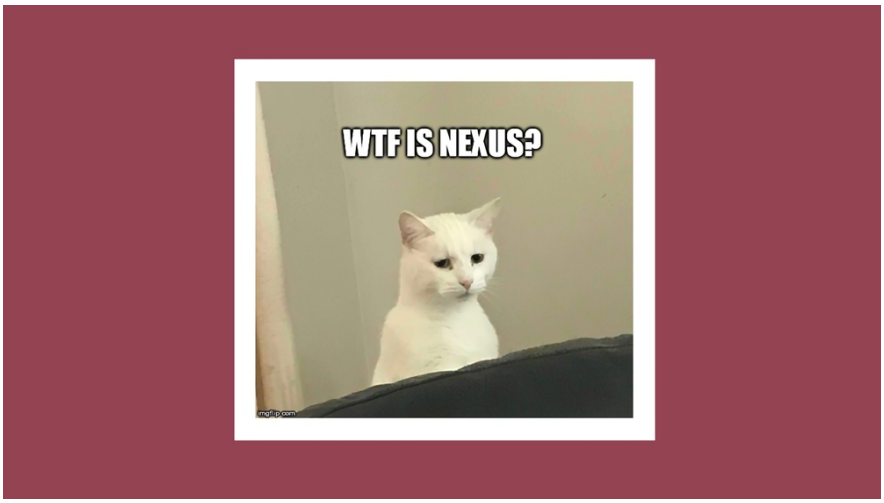
This thesis can be further extended for future work. First, in our work, we mainly tackle the problem of real JPEG restoration by a flexible network using only single compressed synthetic data, providing an insight on improving the generalization ability with limited data. However, in the real-world, images may be corrupted with various kinds of degradation, such as blurring. Designing a more realistic degradation model for training is a promising solution for real application. Second, the performance of restoration is largely influenced by the network architecture. Therefore, designing novel building blocks which are skilled at image restoration is still necessary. Third, compared with traditional methods, learning-based methods enjoy fast inference speed but usually lack flexibility. We solve this problem by using a conditional network. Another promising solution is to compare deep learning with traditional methods to unleash the greatest potential. Fourth, we may try to train a flexible model with unknown quality factors for better real application. Fifth, as we have discussed in this thesis, existing non-reference image quality measures have their own limitations and can not reflect the image quality accurately. Therefore, designing a no-reference image quality measure which can evaluate the images accurately and comprehensively would be of great value for the subsequent research towards real image restoration.

A The Appendix

A.1. More Results on Real JPEG Images



(a) Input



(b) Output

(a) Input

(b) Output

(a) Input

(b) Output

(a) Input

(b) Output

Bibliography

[AB14] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

[AMFM10] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2010.

[AT17] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017.

[BBB+17] Mauro Barni, Luca Bondi, Nicolò Bonettini, Paolo Bestagini, Andrea Costanzo, Marco Maggini, Benedetta Tondi, and Stefano Tubaro. Aligned and non-aligned double jpeg detection using convolutional neural networks. *Journal of Visual Communication and Image Representation*, 49:153–163, 2017.

[BCS10] Mauro Barni, Andrea Costanzo, and Lara Sabatini. Identification of cut & paste tampering by means of double-jpeg detection and image segmentation. In *International Symposium on Circuits and Systems*, pages 1687–1690. IEEE, 2010.

[BLAY14] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pages 226–234. PMLR, 2014.

[BP11] Tiziano Bianchi and Alessandro Piva. Analysis of non-aligned double jpeg artifacts for the localization of image forgeries. In *International Workshop on Information Forensics and Security*, pages 1–6. IEEE, 2011.

[BP12] Tiziano Bianchi and Alessandro Piva. Image forgery localization via block-grained analysis of jpeg artifacts. *IEEE Transactions on Information Forensics and Security*, 7(3):1003–1017, 2012.

[CCK+18] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[CH11] Yi-Lei Chen and Chiou-Ting Hsu. Detecting recompression of jpeg images via periodicity analysis of compression artifacts for tampering detection. *IEEE Transactions on Information Forensics and Security*, 6(2):396–406, 2011.

[CHB17] Lukas Cavigelli, Pascal Hager, and Luca Benini. Cas-cnn: A deep convolutional neural network for image compression artifact suppression. In *International Joint Conference on Neural Networks*, pages 752–759. IEEE, 2017.

[CNL+20] Xuanhong Chen, Bingbing Ni, Naiyuan Liu, Ziang Liu, Yiliu Jiang, Loc Truong, and Qi Tian. Coogan: a memory-efficient framework for high-resolution facial attribute editing. In *European Conference on Computer Vision*, pages 670–686. Springer, 2020.

[CP16] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2016.

[DDCLT15] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *International Conference on Computer Vision*, pages 576–584, 2015.

[DLHT14] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014.

[DO18] Nandita Dalmia and Manish Okade. Robust first quantization matrix estimation based on filtering of recompression artifacts for non-aligned double compressed jpeg images. *Signal Processing: Image Communication*, 61:9–20, 2018.

[ELDS20] Max Ehrlich, Ser-Nam Lim, Larry Davis, and Abhinav Shrivastava. Quantization guided jpeg artifact correction. In *European Conference on Computer Vision*, 2020.

[FSS07] Dongdong Fu, Yun Q Shi, and Wei Su. A generalized benford's law for jpeg coefficients and its applications in image forensics. In *Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, page 65051L. International Society for Optics and Photonics, 2007.

[FZW+19] Xueyang Fu, Zheng-Jun Zha, Feng Wu, Xinghao Ding, and John Paisley. Jpeg artifacts reduction via deep convolutional sparse coding. In *International Conference on Computer Vision*, pages 2501–2510, 2019.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GC16] Jun Guo and Hongyang Chao. Building dual-domain representations for compression artifacts reduction. In *European Conference on Computer Vision*, pages 628–644. Springer, 2016.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, pages 2672–2680, 2014.

[GPBB14] Fausto Galvan, Giovanni Puglisi, Arcangelo Ranieri Bruna, and Sebastiano Battiato. First quantization matrix estimation from double compressed jpeg images. *IEEE Transactions on Information Forensics and Security*, 9(8):1299–1310, 2014.

[GSBDB17] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *IEEE International Conference on Computer Vision*, pages 4826–4835, 2017.

[GYZ+19] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1712–1722, 2019.

[HDQ20] Jingwen He, Chao Dong, and Yu Qiao. Interactive multi-dimension modulation with dynamic controllable residual learning for image restoration. In *European Conference on Computer Vision*. Springer, 2020.

[HZK+19] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE Transactions on Image Processing*, 28(11):5464–5478, 2019.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

[JAFF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[JM18] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *International Conference on Learning Representations*, 2018.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[KSC20] Yoonsik Kim, Jae Woong Soh, and Nam Ik Cho. Agarnet: Adaptively gated jpeg compression artifacts removal network for a wide range quality factor. *IEEE Access*, 8:20160–20170, 2020.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25:1097–1105, 2012.

[KW14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[LDX+19] Ming Liu, Yukang Ding, Min Xia, Xiao Liu, Errui Ding, Wangmeng Zuo, and Shilei Wen. Stgan: A unified selective transfer network for arbitrary image attribute editing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3673–3682, 2019.

[LQHQ07] Weiqi Luo, Zhenhua Qu, Jiwu Huang, and Guoping Qiu. A novel method for detecting cropped and recompressed image block. In *International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages II–217. IEEE, 2007.

[LQLHST19] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. Controllable text-to-image generation. In *Neural Information Processing Systems*, 2019.

[LQLT19] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Controllable text-to-image generation. In *Neural Information Processing Systems*, 2019.

[LQLT20] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. Manigan: Text-guided image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7880–7889, 2020.

[LSH08] Bin Li, Yun Q Shi, and Jiwu Huang. Detecting doubly compressed jpeg images by using mode based first digit features. In *IEEE Workshop on Multimedia Signal Processing*, pages 730–735. IEEE, 2008.

[LSK+17] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 136–144, 2017.

[LTH+17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[LZZ+18] Pengju Liu, Hongzhi Zhang, Kai Zhang, Liang Lin, and Wangmeng Zuo. Multi-level wavelet-cnn for image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2018.

[MFTM01] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, volume 2, pages 416–423. IEEE, 2001.

[MSB12] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2012.

[N+11] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[PBPG14] Cecilia Pasquini, Giulia Boato, and Fernando Pérez-González. Multiple jpeg compression detection by means of benford-fourier coefficients. In *International Workshop on Information Forensics and Security*, pages 113–118. IEEE, 2014.

[PCAL18] Jinseok Park, Donghyeon Cho, Wonhyuk Ahn, and Heung-Kyu Lee. Double jpeg detection in mixed jpeg quality factors using deep convolutional neural network. In *European Conference on Computer Vision*, pages 636–652, 2018.

[PLWZ19] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[RAY+16] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[RSRB15] Gernot Riegler, Samuel Schulter, Matthias Ruther, and Horst Bischof. Conditioned regression models for non-blind single image super-resolution. In *International Conference on Computer Vision*, pages 522–530, 2015.

[RVM+11] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, 2011.

[She05] HR Sheikh. Live image quality assessment database release 2. *http://live. ece. utexas. edu/research/quality*, 2005.

[SZ15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, May 2015.

[TAVG+17] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 114–125, 2017.

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pages 1096–1103, 2008.

[Wal92] Gregory K Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[WFSZ20] Menglu Wang, Xueyang Fu, Zepei Sun, and Zheng-Jun Zha. Jpeg artifacts removal via compression quality ranker-guided networks. In *International Joint Conferences on Artificial Intelligence*, 2020.

[WGTY19] Wei Wang, Ruiming Guo, Yapeng Tian, and Wenming Yang. Cfsnet: Toward a controllable feature space for image restoration. In *International Conference on Computer Vision*, pages 4140–4149, 2019.

[WPLK18] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *European Conference on Computer Vision*, pages 3–19, 2018.

[WYDL18] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 606–615, 2018.

[WYW+18] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *European Conference on Computer Vision*, 2018.

[WZ16] Qing Wang and Rong Zhang. Double jpeg compression forensics based on a convolutional neural network. *EURASIP Journal on Information Security*, 2016(1):1–12, 2016.

[XYL+17] Fei Xue, Ziyi Ye, Wei Lu, Hongmei Liu, and Bin Li. Mse period based estimation of first quantization step in double compressed jpeg images. *Signal Processing: Image Communication*, 57:76–83, 2017.

[XZH+18] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1316–1324, 2018.

[YB10] Changhoon Yim and Alan Conrad Bovik. Quality assessment of deblocked images. *IEEE Transactions on Image Processing*, 20(1):88–98, 2010.

[YHN+16] Liyang Yu, Qi Han, Xiamu Niu, SM Yiu, Junbin Fang, and Ye Zhang. An improved parameter estimation scheme for image modification detection based on dct coefficient analysis. *Forensic Science International*, 259:200–209, 2016.

[YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

[YWQZ20] Heng Yao, Hongbin Wei, Chuan Qin, and Xinpeng Zhang. An improved first quantization matrix estimation for nonaligned double compressed jpeg images. *Signal Processing*, 170:107430, 2020.

[ZCT+19] Bolun Zheng, Yaowu Chen, Xiang Tian, Fan Zhou, and Xuesong Liu. Implicit dual-domain convolutional network for robust color image compression artifact reduction. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[ZEP10] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, pages 711–730. Springer, 2010.

[ZGT20] Kai Zhang, Luc Van Gool, and Radu Timofte. Deep unfolding network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3217–3226, 2020.

[ZLL+18] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *European Conference on Computer Vision*, pages 286–301, 2018.

[ZLL+19] Y Zhang, K Li, K Li, B Zhong, and Y Fu. Residual non-local attention networks for image restoration. In *International Conference on Learning Representations*, 2019.

[ZXL+17] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[ZYHL18] Xiaoshuai Zhang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. Dmcnn: Dual-domain multi-scale convolutional neural network for compression artifacts removal. In *International Conference on Image Processing*, pages 390–394. IEEE, 2018.

[ZZC+17] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[ZZZ18a] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[ZZZ18b] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3262–3271, 2018.

[ZZZ19] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1671–1681, 2019.