Development and Implementation of Deep Learning Algorithms for Restoring Images Degraded by JPEG Compression

Syed Safwan Sajjad Rakib Ahsan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

April 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Syed Safwan Sajjad Rakib Ahsan**

Entitled: **Development and Implementation of Deep Learning Algorithms for Restoring Images Degraded by JPEG Compression**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair
Dr. Yousef R. Shayan

______________________________ Examiner
Dr. Rajamohan Ganesan

______________________________ Examiner
Dr. Chunyan Wang

______________________________ Supervisor
Dr. M. Omair Ahmad

Approved by ______________________________
Dr. Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

April, 2024 ______________________________
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Development and Implementation of Deep Learning Algorithms for Restoring Images Degraded by JPEG Compression

Syed Safwan Sajjad Rakib Ahsan

JPEG is one of the most popular image compression techniques in the world. Its effectiveness has led to it being used in diverse sectors such as satellite imaging, medical imaging, image storage systems and multimedia. With the diverse use of JPEG compression algorithms, it has also become necessary to develop deblocking algorithms to mitigate the compression loss caused by the compression. With the advent of deep learning, several methods have been developed for JPEG image deblocking. The quality factor or QF value is vital to the compression process. Most deep JPEG deblocking networks face the challenge of requiring the image to be compressed by a QF value which is part of the training process. If the image is compressed by any other QF value, the performance and deblocking quality of the network severely degrades.

In this thesis, two different schemes are proposed to solve this issue. The first proposed scheme aims to tackle the problem from the out-of-distribution point of view, whereas the second proposed network aims to tackle the problem from a meta-learning point of view. The effectiveness of the proposed schemes is validated by conducting experiments employing two different benchmark datasets. The proposed networks are shown to outperform the state-of-the-art deep JPEG deblocking networks as shown by the quantitative and qualitative comparative studies.

Acknowledgments

First and foremost, I want to thank Allah, God Almighty, for granting me the strength, perseverance, and ability to complete my thesis.

I am incredibly grateful to my supervisor, Dr. M. Omair Ahmad, for giving me the opportunity to do my thesis under his supervision. Throughout my MASc. journey, he has been highly supportive and ensured I was always on the right track. Without his proper guidance, I could not have completed the work of this thesis.

I thank Dr. Alireza Esmaeilzehi for sharing his suggestions and technical expertise. I could always reach out to him whenever I needed any help.

I thank the Natural Sciences and Engineering Research Council (NSERC) for funding my research.

I am eternally grateful to my loving parents, sister, and wife for always praying for me, believing in me, and supporting me tirelessly.

I met wonderful people during my MASc, and I am incredibly thankful to all of them. I especially want to thank Mr. Azfar Adib for introducing me to machine learning and always being there whenever I needed help or suggestions.

My acknowledgment will not be complete if I do not mention two wonderful people without whom pursuing a degree in a foreign country would have been incredibly difficult. My uncle, Mr. Tanfisur Rahman, and my aunt, Mrs. Farhana Choudhury. My heartfelt thanks to both of them for making Montreal a home away from home and always giving me unconditional love and support.

Contents

List of Figures

List of Tables

List of Abbreviations

ARCNN	Artifacts reduction convolutional neural networks
Adam	Adaptive moment estimation
BRISQUE	Blind/Referenceless image spatial quality evaluator
CNN	Convolutional neural network
DCT	Discrete cosine transform
FC	Fully connected
GAP	Global average pooling
JPEG	Joint photographic expert group
NIQE	Natural Image quality evaluator
PSNR	Peak signal-to-noise ratio
QF	Quality factor
RBQE	Resource-efficient blind quality enhancement
RDN	Residual Dense network
RLE	Run length encoding
RNN	Recurrent neural network
RSTB	Residual swin transformer blocks
ReLU	Rectified linear unit
SSIM	Structural similarity
Swin	Sliding window
TNRD	Trainable nonlinear reaction diffusion

List of Symbols

α, β	Learning rate
Σ	Summation
Θ	Network parameters
C	Number of channels
c	Confidence score
$h(.)$	Image quality assessment module
$\mathcal{L}$	Loss function
$\mathcal{L}^{te}$	Meta-test loss
$\mathcal{L}^{tr}$	Meta-train loss
$o(.)$	Out-of-distribution detection module
$\mathbf{u}$	Probability vector
$\mathbf{v}$	Ground-truth vector
$x[m,n]$	Ground-truth image
$\hat{x}[m,n]$	High quality deblocked image
$y_q[m,n]$	Input image

Chapter 1

Introduction

1.1 General

With the advent of digital cameras and the explosion of digital images on the internet, image compression has become an extremely essential task. Various image compression algorithms have been developed for the efficient use of storage systems for images. JPEG is one of the most popular and widely used algorithms in image compression. JPEG finds its application in various fields, including satellite imaging, medical imaging, image storage systems, and multimedia.

One of the main drawbacks of using JPEG for image compression is that JPEG is a lossy image compression technique. It means that decompressing an image compressed by the JPEG algorithms does not result in retrieving the original image. The decompressed image suffers from a lack of visual quality and often has noticeable deformations caused by the suppression of high-frequency components in the image, commonly known as blocking artifacts. The main goal of developing the state-of-the-art decompression algorithm is to make the compressed image more visually appealing and reduce the compression artifact.

Before the advent of deep-learning methods, traditional methods were predominantly used for the task of JPEG image deblocking. Adaptive post-filtering [1], pointwise shape-adaptive DCT (SA-DCT) [2] and wavelet-based deblocking [3] are some of the widely used traditional techniques. Among these traditional techniques SA-DCT [2] has shown the best performance. One of the earliest deep learning based method ARCNN [4] has succesfully outperformed the best performing classical method SA-DCT. Afterwards, the research community focused on developing newer and better performing deep learning based methods for the task of JPEG image deblocking rather than focusing on the traditional techniques.

Deep neural networks have shown remarkable results in various image restoration tasks [5–21]. They have also performed impressively when applied to deblocking images compressed by the JPEG algorithm [4], [22], [23], [24], [25]. By optimizing a non-linear function using backpropagation, deep neural networks learn the mapping between the degraded image and their corresponding high-quality images. The research community has developed a lot of deep neural network-based JPEG Deblocking networks, which can mainly be classified into Deep Non-blind JPEG Deblocking and Deep Blind JPEG Deblocking networks.

Non-blind JPEG Image deblocking networks focus on removing the artifacts of a degraded image decompressed by a unique and known QF value. One of the first networks of such a kind is the artifact reduction convolutional neural network (ARCNN) [4]. The deblocking of the image is done in four stages in ARCNN, namely feature extraction, feature enhancement, mapping, and reconstruction. ARCNN shows improved performance from its predecessors, such as SRCNN [26], by improving the mapping accuracy by deploying two layers of feature extraction and enhancing the extracted low-level features.

In Trainable nonlinear reaction diffusion (TNRD) [22], diffusion partial differential equation is used to model the task of JPEG image deblocking. The solution of the partial differential equation is then implemented using convolution operation and non-linear activation functions. The parameters are obtained by training the network with the degraded image and their corresponding ground truth image using backpropagation.

Deepboosting [27] is a network that combines the boosting algorithm with convolutional neural networks to improve the performance of various image restoration tasks, including JPEG image deblocking. Deepboosting uses higher depth and width to provide significant improvement in performance. The vanishing gradient problem is dealt with by deploying dense residual connections.

A significant drawback of using deep non-blind JPEG deblocking networks is that their performance degrades severely when applied to deblock degraded images compressed by QF values other than the values used to train the network. To address this issue, deep non-blind networks have been developed by the research community. Deep blind networks are discussed in Section 1.2.

1.2 A Brief Review of Deep Blind JPEG Image Deblocking Networks

Deep blind networks deal with the problem of deblocking images degraded by a value of the quality factor not used in the images for the training of the networks. A special type of deep blind network can deal with a range of QF values using multiple models of the same network trained with images compressed with different QF values. These networks are called deep pseudo-blind networks. In this section, a review of some of the deep pseudo-blind JPEG image deblocking networks [28–31] is presented.

Most deep pseudo-blind networks have two parts. The first part, known as the QF estimation module, estimates the probability of a QF used to compress the input image, and the second part is a collection of a certain number of pre-trained models of a deep network that actually performs the task of deblocking. A QF estimation module is used to estimate the probability that the QF value of the input image falls into certain classes for which an equal number of models of the same network have already been trained. For example, if the pseudo-blind scheme in its second part has four models of the network pre-trained with images with QF values of q_1, q_2, q_3 and q_4, respectively, then the QF estimation module estimates the probabilities that the input image has been degraded using QFs of q_1, q_2, q_3 and q_4. Then, in the second part, the network trained using the images degraded with the QF with the highest probability is selected to deblock the input image.

In the pseudo-blind network of [28], the authors have devised a deep dual domain blind network (D3SN). The deblocking network (the second part) consists of two parallel segments: a sub-network that performs deblocking in the DCT domain and a sub-network that performs deblocking in the pixel domain. The outputs of the two networks are then suitably combined to get the final deblocked image.

In [29], inception-based artifact reduction convolution neural network (IACNN) is one of the first networks to use inception blocks for JPEG image deblocking. It uses a cascade of two inception blocks for the task of blind JPEG image deblocking. The use of inception blocks enables the network to deblock the compressed image by extracting feature maps at various depths and extracting multi-scale features.

Another notable network for blind JPEG image deblocking is the dual pixel wavelet domain with a soft encoding network (DPW-SDNet) [30]. This combines the prediction from the pixel and wavelet domains to provide soft decoding performance. The combination of the outputs of the two branches of DPW-SDNet, the pixel domain branch and the wavelet domain branch, results in the extraction of better feature maps for the tasks of JPEG image deblocking.

The resource-efficient blind quality enhancement (RBQE) scheme [31] is another deep blind JPEG image deblocking network that increases the depth of the deblocking network based on the estimated QF of the input degraded image. When the QF is lower, the compression noise is higher. Hence, a deeper network is required to deblock the compression artifacts. To determine the stopping point of applying the deblocking operation, RBQE utilizes the sum of squared non-DC Tchebichef moments for performing the non-reference-based image quality assessment.

1.3 Motivation and Objectives

As seen from the review of Section. 1.2, the deep pseudo-blind networks succeed in solving the problem of deblocking an image in which the degradation QF of the input image is unknown only to a certain extent. In an ideal scenario, if the deblocking network has four models trained with QF values, q1, q2, q3 and q4, and if the input image has been compressed by one of these four QFs, then such networks will result in deblocking the input image successfully. However, if the QF of the input image is not one of these four QF values, then the performance may be severely affected.

The objective of this thesis is to develop deep JPEG deblocking schemes that can address the drawbacks of the existing deep pseudo-blind JPEG deblocking networks mentioned above. This thesis proposes two different schemes for blind JPEG image deblocking. In the first scheme, a novel deep learning-based blind JPEG image deblocking scheme is developed that utilizes four modules, namely, JPEG image QF detection, out-of-distribution detection, image quality assessment, and JPEG image deblocking module, which can efficiently handle the degraded decompressed images obtained by in and out-of-distribution QF values. In the JPEG deblocking module of this scheme, there are multiple models of the same deep deblocking network, each of which is trained with a particular QF value. In the second scheme, a single model of a JPEG deblocking network is used. The model is trained using a meta-learning algorithm to make it robust enough to deal with images degraded with a wider range of QF values compared to the first scheme thus improving the network's generalization capability. In view of the fact that in this scheme uses only one trained model, its computational complexity and memory usage is significantly lower than that of the first scheme with a slightly lower performance.

1.4 Organization of the Thesis

The thesis is organized as follows. Chapter 2 provides some background material, such as JPEG compression, convolutional neural networks, transformers and SwinIR architecture, pertinent to the work undertaken in this thesis. Chapter 3 starts with the presentation of the first JPEG deblocking scheme proposed in this thesis. This scheme, referred to as OOD-Net, is a deep blind JPEG image compression deblocking scheme that utilizes in and out-of-distribution detection. A detailed ablation study is carried out on the proposed scheme to show the effectiveness of the design of the proposed scheme. Performance of the proposed scheme is compared with that of the existing state-of-the-art schemes. In Chapter 4, the second scheme, referred to as MetaNet , for JPEG image deblocking using meta-learning is presented. A detailed ablation study is performed to demonstrate the importance of the various ideas used in the proposed scheme. The performance of the proposed scheme, in terms of PSNR and SSIM, is compared with the state-of-the-art schemes. Finally, Chapter 5 concludes the thesis by summarizing the work and highlighting the contributions made in this thesis. Some suggestions for further investigation of the work undertaken in the thesis are also provided.

Chapter 2

Background Material

In this chapter, some background materials pertinent to the work undertaken in this thesis are presented. First, a brief overview of the JPEG image compression algorithm is given. Then, a brief discussion on convolutional neural networks and transformers is given based on which the state-of-the-art deep blind and non-blind JPEG image deblocking networks are built. Finally, a detailed discussion on the network of SwinIR [32] is presented based on which OODNet in Chapter 3 and MetaNet in Chapter 4 is designed. All these aforementioned topics are vital as background materials to understand the work presented in this thesis.

2.1 JPEG Image Compression Algorithm

Since its standardization in 1992, JPEG has been the most popular image compression algorithm for reducing the size of digital images while maintaining perceptual quality. JPEG is a lossy compression technique, which means that information is lost during this process. To understand the process further, we discuss the JPEG image compression algorithm for RGB images in this section.

The first step towards compressing an RGB image is transforming the color space to

YCbCr. YCbCr also has three channels; the Y channel is the luminance channel containing the brightness information, whereas Cb and Cr contain color information of the chrominance blue and chrominance red channels. Human eyes are more sensitive to the brightness of an image or the luminance channel. The JPEG algorithm exploits this phenomenon by reducing information from the luminance channels without significantly impacting the perceptual quality.

The second step is downsampling. As discussed in the previous paragraph, human eyes are not very sensitive to the color information of an image. Hence, downsampling is applied on the chrominance channels Cb and Cr. The luminance channel is kept intact at this stage of the algorithm. The application of downsampling on the chrominance channels reduces the image size to half its original size.

Then, in the third step, the pixel data of each of the three channels is divided into blocks of size 8X8. After this, the later mathematical operations are applied to each pixel individually.

Next, each of the aforementioned blocks of size 8X8 is passed through a two-dimensional DCT (2D-DCT) transformation. Let $x[m,n]$ be a high-quality image ($0 \leq m \leq M-1, 0 \leq N \leq N-1$), the size of which we want to compress. The image $x[m,n]$ is divided into R non-overlapping blocks $w_r[m,n]$ ($r = 1, ..., R$), each of which has a spatial resolution of size 8X8. After that, each block $w_r[m,n]$ is passed through a two-dimensional DCT (2D-DCT) transformation, and we obtain its coefficients $W_r[k,l]$ as:

$$\begin{aligned} W_r[k,l] = 4\sum_{m=0}^{T-1}\sum_{n=0}^{T-1} w_r[m,n]\cos(\frac{k\pi(2m+1)}{2T}) \\ \cos(\frac{l\pi(2n+1)}{2T}) \qquad 0 \leq k \leq T-1, 0 \leq l \leq T-1 \end{aligned} \tag{1}$$

The next step is quantization. To do so, the matrix obtained by applying 2D-DCT is divided by a precalculated matrix, and the results are rounded to the nearest integer value.

The precalculated matrix or quantization table is generated using the quality factor (QF) value. The higher the QF, the lower the compression and the higher the image quality. QF can be any value between 1 to 100, where 1 means the highest compression and 100 means no compression. Hence, we can say that QF value plays a significant role in the compression process. A quantization matrix generated by MATLAB for QF = 30 is given in Fig. 2.1.

$$\begin{bmatrix}
27 & 18 & 17 & 27 & 40 & 67 & 85 & 102 \\
20 & 20 & 23 & 32 & 43 & 97 & 100 & 92 \\
23 & 22 & 27 & 40 & 67 & 95 & 115 & 93 \\
23 & 28 & 37 & 48 & 85 & 145 & 133 & 103 \\
30 & 37 & 62 & 93 & 113 & 182 & 172 & 128 \\
40 & 58 & 92 & 107 & 135 & 173 & 188 & 153 \\
82 & 107 & 130 & 145 & 172 & 202 & 200 & 168 \\
120 & 153 & 158 & 163 & 187 & 167 & 172 & 165
\end{bmatrix}$$

Figure 2.1: Quantization Matrix for Q=30 generated using MATLAB.

We can see from the quantization matrix that the values are higher in the bottom right portion of the matrix, which signifies that high-frequency components of the image are suppressed more compared to the lower frequencies. This works because human eyes are less sensitive to the high-frequency components of the image.

The final step in JPEG image compression is entropy encoding. This step is completely lossless as no information is lost. RLE (Run Length Encoding) and Huffman Coding algorithm are used to reduce the space without losing any information.

The JPEG compression algorithm is depicted in Fig. 2.2, where we can clearly see the six steps involved in compressing an RGB image.

When the compressed image is decoded by an image viewer, all the steps mentioned above are performed in the reverse order. The compression artifacts are a result of applying DCT to the image to reduce the size of the image. The goals of JPEG image deblocking is

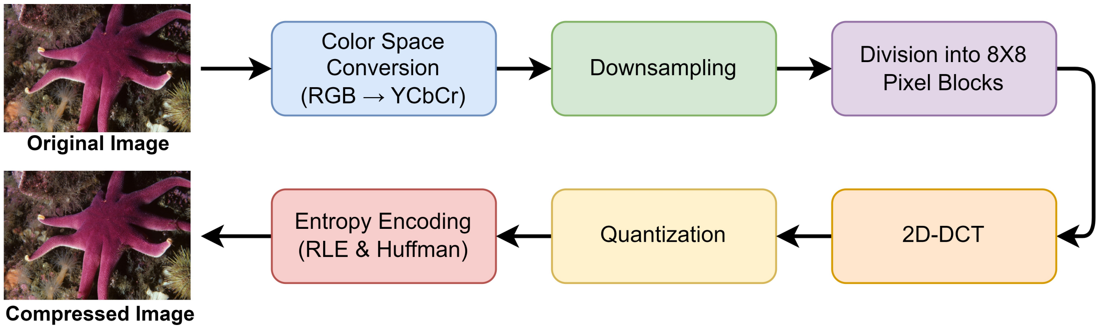

Original Image
Color Space Conversion (RGB → YCbCr)
Downsampling
Division into 8X8 Pixel Blocks
2D-DCT
Quantization
Entropy Encoding (RLE & Huffman)
Compressed Image

Figure 2.2: Graphical Illustration of the JPEG Compression Algorithm

to remove the blocking artifacts created by the application of DCT in images.

2.2 Convolutional Neural Network

Convolutional neural networks (CNN) have revolutionized deep learning. It has been applied successfully to various computer vision and natural language processing tasks. CNN was first introduced in 1998 by LeCun, et.al [33], which solved a practical problem of recognizing hand-written digits. The network was called LeNet, which was named after LeCun. With the advancement of computer hardware and the availability of GPUs capable of parallel computing, increasing the depth of convolutional neural networks has been possible. Some examples of deep CNNs include VGG16 [34] and ResNet [35]. Deep CNN has been successfully used in various vision tasks including JPEG image deblocking [4], [29], image superresolution [26], [36], [37], [38] and image classification [35], [34]. A convolutional neural network capable of image classification has three layers: convolutional layers, pooling layers, and fully connected layers. Each of the layers plays a vital role in the network and has specific tasks associated with them.

The convolutional layers apply the mathematical operation on the intended image to extract feature maps. This layer deploys multiple filters of size D X H X W, where D is the depth of the filters, and H and W are, respectively, the height and width of the filters.

Activation functions such as ReLU [39] are added after the convolution layer to introduce non-linearity in the network.

Pooling layers are another vital part of the convolutional neural network. Pooling layers reduce the spatial resolution of the feature maps, which aids significantly in reducing the computational complexity. The decrease in spatial resolution enables further use of a larger number of filters to extract more robust features without increasing the computational complexity. There are two main types of pooling operations: average pooling and max pooling, which are used based on the design and requirements of the network.

The final layer of a typical convolutional neural network architecture is fully connected layers. Fully connected layers are mainly used for image classification, image clustering, and speech recognition tasks. For other tasks, such as image denoising and image deblocking, the final layer is a $1X1$ convolutional layer. For classification purposes, the output of the final fully connected layer is passed through a SoftMax function to get a probability distribution from the output.

We discussed the components in a typical convolutional neural network above. Along with the components mentioned above, various other operations, such as batch normalization and layer normalization, are also applied based on the type and requirement of the network. A typical convolutional neural network is depicted in figure reffig:.

2.3 Transformer

Transformer, first introduced in [40], has revolutionized the fields of natural language processing and computer vision. The ability to handle sequential data made transformers useful for natural language processing tasks. With the advent of Vision Transformers (ViT) [41], transformers have been successfully utilized in computer vision tasks.

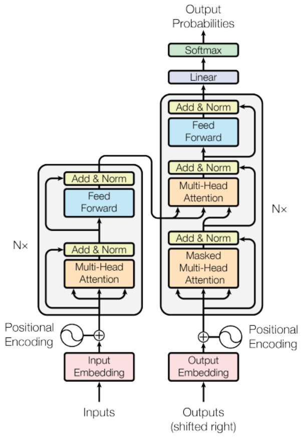


Figure 2.3: Basic Architecture of a Transformer

This section discusses the transformer architecture introduced in [40]. The transformer mainly has two parts, namely- encoder and decoder.

The encoder is composed of six identical layers stacked upon each other. Each layer consists of two sub-layers, a multi-head self-attention mechanism, and a second is a simple position-wise fully connected feed-forward network. A residual connection [35] followed by a layer normalization [42] is employed around each sub-layer.

The decoder also is composed of six identical layers like the encoder. There is an additional third sub-layer of multi-head attention along with the previous two sub-layers, multi-head self-attention, and a simple position-wise fully connected feed-forward network of the encoder. Residual connection and layer normalization are also applied. The self-attention sub-layer in the decoder stack is also modified to aid in better prediction results.

The basic architecture of the transformer is depicted in Fig. 2.3. This figure is taken from [40], where transformers were first introduced.

2.4 SwinIR:Image Restoration Using Swin Transformer

SwinIR [32] is a state-of-the-art network that has successfully been applied to image super-resolution, image denoising, and JPEG image deblocking. SwinIR combines the capabilities of both convolutional neural networks and transformers. The main block of SwinIR is the Swin transformer layer [43]. In this section we discuss about the details of the SwinIR network.

SwinIR architecture can be divided into three functional parts: shallow feature extraction, deep feature extraction, and high-quality image reconstruction. Each of these parts plays a unique role, and together, they provide superior performance in image restoration tasks.

Shallow feature extraction is built using convolutional layers. This layer extracts feature maps, which later help in deep feature extraction. As convolution layers are very good at early visual processing, it leads to stable training and optimization of the SwinIR network. A 3 X 3 convolutional layer is used to extract the feature maps.

Deep feature extraction utilizes the capacity of Swin Transformer layers, which are embedded within the RSTB (Residual Swin Transformer Block) with residual connections. Residual connections have been used a lot to improve the performance of various computer vision tasks, such as image classification [35], image superresolution [44], [45] [46], and JPEG image deblocking and denoising [47]. The output of the final STL of a RSTB is passed through a 3 X 3 convolutional layer. Adding this final layer of convolution at the end of an RSTB helps in an efficient combination of the shallow and deep features.

The final part of the SwinIR network is the high-quality image reconstruction module. This module does the final combination of the shallow and deep features. Shallow features mainly contain low-frequency components, whereas deep features extracted by the deep

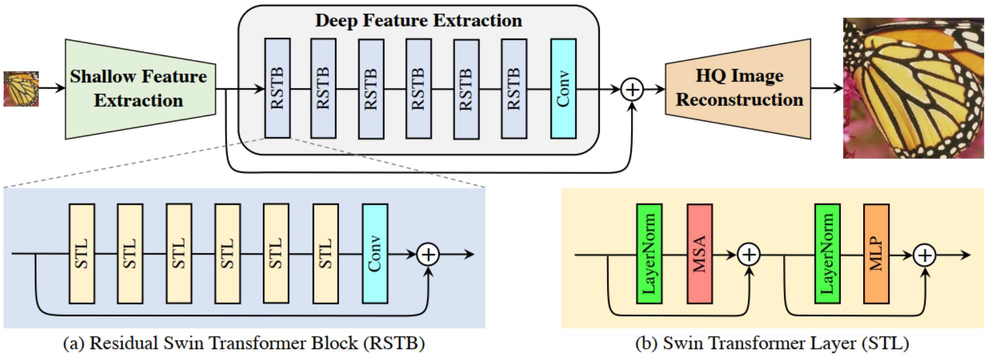


Figure 2.4: Network Architecture of SwinIR

feature extraction module contain high-frequency components. The low-frequency components extracted by the shallow feature extraction module are directly transferred to the image reconstruction module using a skip connection. For the task of JPEG image deblocking, which does not require upsampling, a single convolutional layer is used for the image reconstruction.

The network architecture of SwinIR is depicted in Fig. 2.4. This figure is taken from [32], the paper introducing SwinIR.

2.5 Summary

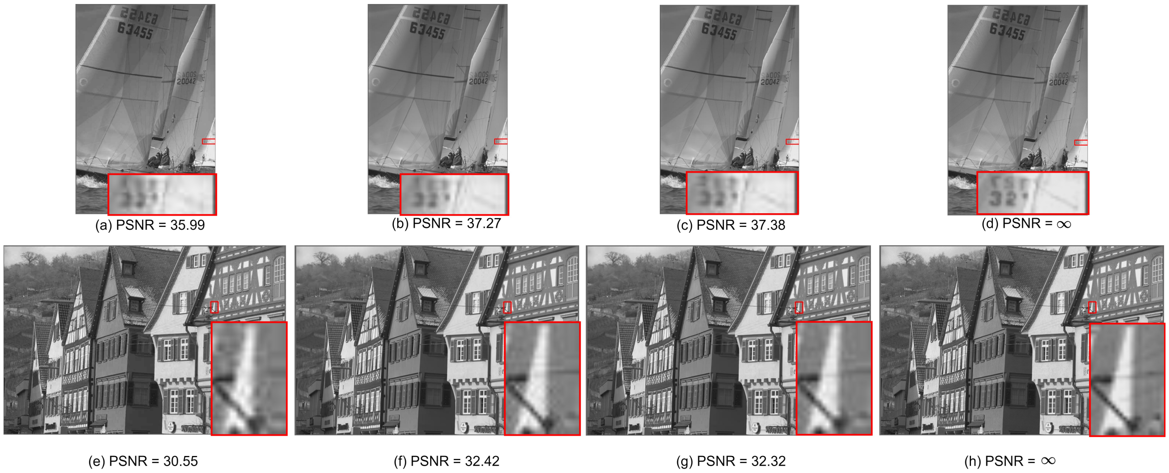
This chapter has presented the relevant background material necessary for understanding the work of this thesis. The chapter starts with a discussion of the JPEG compression algorithm. Gradually, the architectures of convolutional neural networks and transformers are discussed. Finally, the chapter is ended with the discussion on SwinIR network, its architecture and various components associated with it.

Chapter 3

JPEG Image Deblocking using In and Out-of-Distribution Detection

3.1 Introduction

Existing state-of-the-art deep pseudo-blind JPEG deblocking networks focus on dealing with QF values of a particular set. For example, most can deal with JPEG images compressed with certain QF values. The performance immediately drops if the image is compressed with any QF value outside the values the networks were used to train with. In an ideal scenario, $q_{max} - q_{min} + 1$ networks can be trained to address this issue where $q_{min} = 1$ and $q_{max} = 100$. This creates a problem as training deep JPEG deblocking networks requires a lot of time and resources. This seriously inhibits the practical usability of such networks. Different images compressed with the same QF factor show variety in performance when passed through networks trained with different QF values. Hence, selecting the appropriate network is of utmost importance for each input image. We can see in Fig. 3.1 that two images compressed with the same QF 35 value, when exposed to two networks trained with QF 30 and QF 40, show different results. For the buildings image, the performance of the deep deblocking network trained with QF = 30 is better, whereas for the sailing3 image the performance of the deep deblocking network trained with QF = 40 is better. In this chapter, we propose OODNet [48] to address the issues and limitations of the existing JPEG deep pseudo-blind image deblocking networks by incorporating in and out-of-distribution detection and providing two different techniques to deal with images compressed with in and out-of-distribution QF values respectively. In section 3.2, the architecture of the proposed OODNet is described. The experimental results and comparative studies are presented in Section. 3.3. Finally, the conclusion of this study is given in Section 3.4.



(a) PSNR = 35.99 (b) PSNR = 37.27 (c) PSNR = 37.38 (d) PSNR = ∞

(e) PSNR = 30.55 (f) PSNR = 32.42 (g) PSNR = 32.32 (h) PSNR = ∞

Figure 3.1: Visual quality of the images degraded by out-of-distribution QF value and restored by SwinIR trained with in-distribution QF values. (a) Degraded image with q = 35. (b) SwinIR trained with q = 30. (c) SwinIR trained with q = 40. (d) Ground truth. (e) Degraded image with q = 35. (f) SwinIR trained with q = 30. (g) SwinIR trained with q = 40. (h) Ground truth.

3.2 Proposed Scheme of OODNet

The overall architecture of OODNet consists of four modules, as seen in Fig. 3.2. The four modules are the image QF detection module, the out-of-distribution module, the JPEG image deblocking module, and the image quality assessment module. In the following subsections, we discuss each of the modules in detail.

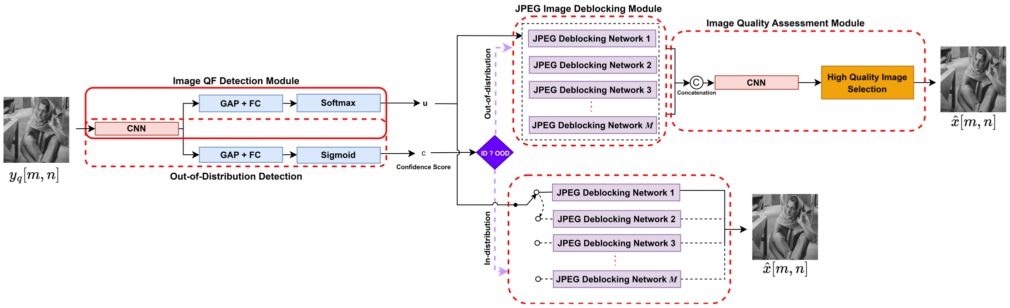

JPEG Image Deblocking Module
Image Quality Assessment Module
JPEG Deblocking Network 1
JPEG Deblocking Network 2
JPEG Deblocking Network 3
JPEG Deblocking Network M
Concatenation
CNN
High Quality Image Selection
$\hat{x}[m,n]$
Image QF Detection Module
CNN
GAP + FC
Softmax
u
GAP + FC
Sigmoid
c
Confidence Score
Out-of-Distribution Detection
$y_q[m,n]$
ID ? OOD
Out-of-distribution
In-distribution
JPEG Deblocking Network 1
JPEG Deblocking Network 2
JPEG Deblocking Network 3
JPEG Deblocking Network M
$\hat{x}[m,n]$

Figure 3.2: Overall architecture of the proposed deep learning-based blind JPEG image deblocking scheme

3.2.1 Image QF detection module

To perform the task of blind JPEG image deblocking, obtaining the QF value q of the input degraded decompressed image $y_q[m,n]$ is vital. The QF value, q, can be chosen as an integer within a particular range $[q_{min}, q_{max}]$ based on the intended application. The deep learning-based blind JPEG image deblocking schemes utilize I networks, each of which is trained for enhancing the quality of the images with a specific QF value q chosen from a set in-distribution QF values $q = \{q_1, q_2, ..., q_M\}$. In-distribution QF values mean the QF values used for training the network. In the image QF detection module of the proposed OODNet, the goal is to assign a QF value from a set of in-distribution QF values to the input degraded image $y_q[m,n]$. For this purpose, a shallow convolutional neural network $f(.)$ is used, which receives the input degraded image $y_q[m,n]$ and outputs its corresponding QF value from the set of in-distribution QF values $q = \{q_1, q_2, ..., q_M\}$.

The architecture of network $f(.)$ consists of a cascade of 4 convolution operations each using 64 filters with kernel size 3×3 and followed by a ReLU [39] activation function. The feature tensor output from the fourth convolution operation of the network $f(.)$ is made to undergo the global average pooling, fully-connected operation with I number of units and a softmax activation, and the probability that the input image is assigned to each of the I in-distribution QF values is yielded. This process is formulated as follows:

$$\mathbf{u} = f(y_q[m, n]) \tag{1}$$

where **u** demonstrates the vector of size $I \times 1$ showing the probability of different in-distribution QF values.

3.2.2 Out-of-distribution Detection Module

When the input image $y_q[m, n]$ is obtained by the in-distribution QF values from the set $q = \{q_1, q_2, ..., q_M\}$, the network $f(.)$ for Image QF detection can confidently estimate its corresponding QF value. On the other hand, when $y_q[m, n]$ is produced by out-of-distribution QF values, none of the I in-distribution QF values is exactly associated with it. The out-of-distribution detection module is developed and incorporated into the proposed scheme by considering this critical point and discriminating between the images obtained by the in-distribution QF values and those yielded by the out-of-distribution QF values.

The role of the out-of-distribution detection module is to determine whether the input degraded image $y_q[m, n]$ is obtained by the in-distribution or out-of-distribution QF values. In this regard, an extra branch is added to the image QF detection module, which predicts the confidence score c associated with the vector **u**, i.e., the probability vector of various in-distribution QF values. The architecture of this extra branch is shown in Fig. 3.2. As seen from this figure, the out-of-distribution detection module consists of a cascade of a global average pooling, fully-connected operation with a single hidden unit, and a sigmoid activation function and outputs the confidence score $c = o(y_q[m,n])$ associated with the estimated probability vector **u**.

Based on the explanations given in the above paragraph, it can be concluded that the image QF detection and out-of-distribution detection modules of the proposed scheme should be trained jointly. In this regard, we train the two modules simultaneously using the following loss function:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}_{f(.)}, \boldsymbol{\Theta}_{o(.)}) = \min_{f(.),o(.)} \Big\| & o(y_q[m,n]) f(y_q[m,n]) + \\ & \big(1 - o(y_q[m,n])\big)\mathbf{v} - \mathbf{v}\Big\|_1^1 - \lambda \log\big(o(y_q[m,n])\big) \\ = \min_{f(.),o(.)} \Big\| & c\,\mathbf{u} + (1-c)\mathbf{v} - \mathbf{v}\Big\|_1^1 - \lambda \log(c)\end{aligned} \tag{2}$$

where $\boldsymbol{\Theta}_{f(.)}$ and $\boldsymbol{\Theta}_{o(.)}$, respectively, denote the parameters of the image QF detection and out-of-distribution detection modules, **v** is the one-hot ground truth vector corresponding to the QF value of the input image $y_q[m,n]$, and λ balances between the two loss terms. λ is set empirically as 10^{-3}. The first term in the loss function shown in (2) strives to provide higher values of c's for the vectors **u**'s that are close to the ground truths **v**'s and generate the smaller values of c's for the vectors **u**'s that are far from their corresponding **v**'s. To avoid obtaining a trivial solution of $c = 0$, the second loss term, which maximizes the entropy of the confidence score c, is employed. The confidence score is an appropriate metric for discriminating between the degraded images $y_q[m,n]$'s obtained by the in-distribution and out-of-distribution QF values. Specifically, the value of the confidence score c would be smaller for the images obtained by the out-of-distribution QF in comparison to those produced by the in-distribution QF. Hence, after the training process of the out-of-distribution detection module $o(.)$ is converged, it is applied to the images of the training dataset that are randomly produced by the JPEG compression technique with various QF values within the range $[q_{min}, q_{max}]$. Next, the confidence score values thus obtained are observed for the images corresponding to the in-distribution QFs and those associated with the out-of-distribution QFs to determine the threshold α for discriminating these images. In the inference stage, the image QF detection and out-of-distribution detection modules are applied to the input degraded decompressed images and their estimated QF values and the confidence scores c's of the estimated QF values are obtained. When the confidence score passes the threshold α, the input image is discerned to be obtained by the estimated in-distribution QF value. Otherwise, it is determined that even though the estimated QF value of the input image is close to its original value, they are not the same.

3.2.3 JPEG Image Deblocking and Image Quality Assessment Modules

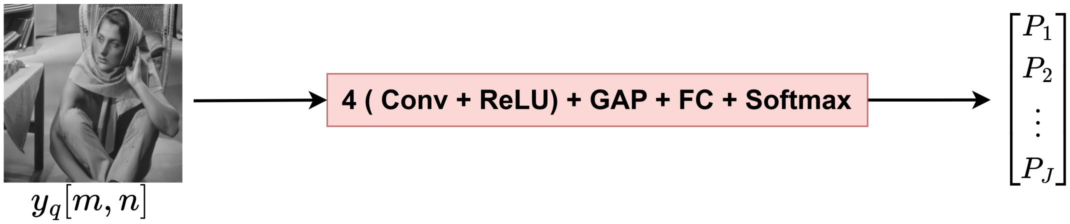
Based on the outcomes of the image QF detection and out-of-distribution detection modules, OODNet performs the deblocking operation differently. To delineate more, when the out-of-distribution detection module determines that the input image $y_q[m,n]$ is obtained by an in-distribution QF value $q = \{q_1, q_2, ..., q_M\}$, our scheme directly employs the JPEG image deblocking module that utilizes a deep network trained with the estimated in-distribution QF value q. The JPEG image deblocking module of the proposed scheme consists of I deep networks, each trained for a specific in-distribution QF value chosen from the set $q = \{q_1, q_2, ..., q_M\}$. Without loss of generality, SwinIR [32] is modified to be further adapted to the task of JPEG image deblocking and used as the architecture of each deep network employed in the JPEG image deblocking module. The modified version of SwinIR and its training details are explained later in this subsection. On the other hand, when the out-of-distribution detection module infers that the input image $y_q[m,n]$ is generated by an out-of-distribution QF value, the proposed scheme utilizes a cascade of JPEG image deblocking module and the image quality assessment module.

As mentioned in Section. 3.1 and shown in Fig. 3.1, in the case that the degraded decompressed images are produced by the out-of-distribution QF values, selecting a suitable deep deblocking network is highly related to the input image contents and textures. Hence, it is paramount to develop an image quality assessment module for every input degraded image obtained by the out-of-distribution QF values to dynamically selects which deep deblocking network results in generating the deblocked image with a higher visual quality. Given these explanations, the training process of the image quality assessment module of the proposed scheme is explained.

Let the image QF detection module assigns the QF value q_i from the set $q = \{q_1, q_2, ..., q_M\}$ to the input degraded image $y_q[m, n]$. It is mentioned above that when the out-of-distribution detection module identifies that the input degraded image $y_q[m, n]$ is obtained by the out-of-distribution QF values, the proposed method employs the image quality assessment module, along with the JPEG image deblocking module. In this case, for training the image quality assessment module, we feed the degraded images obtained by the out-of-distribution QF values to the two deep networks $g_j(.)$, $j = 1, 2$ from the JPEG image deblocking module, which is already trained with the QF values q_i and q_{i+1}, respectively, and obtain the two deblocked images $g_j(y_q[m, n])$, $j = 1, 2$. Next, the ℓ_1 norm losses between these two images and the ground truth signal $x[m, n]$ are obtained as:

$$P_j = \Big\|g_j(y_q[m, n]) - x[m, n]\Big\|_1 \quad (j = 1, 2) \tag{3}$$

The index j of the deep network that leads to a lower value of P_j is considered the image quality assessment label of the input degraded image $y_q[m, n]$. Finally, a shallow convolutional network, $h(.)$, that is shown in Fig. 3.3 is trained, to map the concatenation of the images $g_j(y_q[m, n])$, $j = 1, 2$, to the image quality assessment label j. This label indicates which of the two deep networks results in generating a deblocked image with superior visual quality. It is seen from Fig. 3.3 that the image quality assessment network, $h(.)$, consists of a cascade of four convolution operations, each using 64 filters with kernel size 3×3, a global average pooling, a fully-connected connection operation using 3 hidden units, and a softmax activation function.



$y_q[m, n]$
4 (Conv + ReLU) + GAP + FC + Softmax
$\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_J \end{bmatrix}$

Figure 3.3: Architecture of the convolutional network used in the image quality assessment module of the proposed scheme.

3.2.4 Network Architecture of the Deep JPEG Image Deblocking Network in OODNet

The architecture of each deep network utilized in the JPEG image deblocking module of the proposed scheme is described in this section. As mentioned in the previous section, the JPEG image deblocking module of the proposed scheme consists of I deep networks, each of which is specified for a single in-distribution QF value chosen from the set $q = \{q_1, q_2, ..., q_M\}$. The network architecture of each of these networks is illustrated in Fig. 3.4. It is seen from this figure that the network architecture of SwinIR [32] is modified, which is generally designed for different tasks of image restoration, to be more adapted for the task of JPEG image deblocking. Like SwinIR, the modified deep JPEG image deblocking network consists of three stages: shallow feature extraction, deep feature extraction, and high-quality image reconstruction. The shallow feature extraction stage is formed from a convolution operation with 180 filters of kernel sizes 3×3. The deep feature extraction module is constructed from 6 residual Swin transformer blocks (RSTBs), each of which is composed of a cascade of 6 Swin transformer layers [32] with 180 embedding dimensions, followed by a convolution operation with 180 filters of kernel size 3×3.

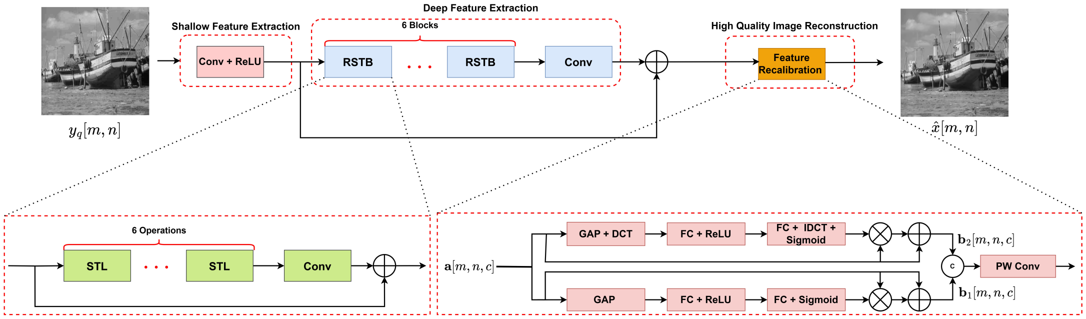


Figure 3.4: Architecture of deep network employed in the JPEG image deblocking module of the proposed scheme

In the high-quality image reconstruction stage of the proposed scheme, a novel operation referred to as the feature recalibration operation is used. The architecture of this modified SwinIR network is shown in Fig. 3.4. As seen from this figure, the proposed feature recalibration operation constitutes two parallel branches; the first one obtains the channel-wise interactions of the features in the spatial domain, and the second one extracts these interactions in the DCT domain. Specifically, let $\mathbf{a}[m, n, c]$ denote the feature tensor of size $M \times N \times C$ input to the feature recalibration operation, where C indicates the number of channels. In the first branch, the feature tensor $\mathbf{a}[m, n, c]$ is passed through the cascade of global average pooling and two fully-connected operations to obtain the recalibration weights $\mathbf{w}_1$ in the spatial domain as:

$$\mathbf{w}_1 = \sigma_2\big(\sigma_1\big(GAP(\mathbf{a}[m, n, c])\big)\big) \tag{4}$$

where $GAP(.)$, σ_1 and σ_2, respectively, denote the global average pooling, fully connected operation with 180 hidden units and the ReLU activation function, and the fully-connected operation with 45 hidden units and the sigmoid activation function. The recalibration weights thus obtained are then multiplied by the input feature tensor $\mathbf{a}[m, n, c]$ yielding the recalibrated feature tensor $\mathbf{b}_1[m, n, c]$ as:

$$\mathbf{b}_1[m, n, c] = (\mathbf{w}_1 \odot \mathbf{a}[m, n, c]) + \mathbf{a}[m, n, c] \tag{5}$$

where $\odot$ denotes the Hadamard product operation. In the second branch, the recalibration weights $\mathbf{w}_2$ are obtained in the DCT domain to emphasize further the features that are useful for the task of JPEG image deblocking. Specifically, the recalibration weights $\mathbf{w}_2$ in the DCT domain are obtained as:

$$\mathbf{w}_2 = \sigma_4\Big(IDCT\Big(\sigma_3\big(DCT\big(GAP(\mathbf{a}[m, n, c])\big)\big)\Big)\Big) \tag{6}$$

where σ_3 and σ_4, respectively, show the fully-connected operation with 180 hidden units and the ReLU activation function and the fully-connected operation with 45 hidden units and the sigmoid activation function. The recalibration weights $\mathbf{w}_2$ thus generated are then multiplied by the feature tensor $\mathbf{a}[m, n, c]$ in order to produce the recalibrated feature tensor $\mathbf{b}_2[m, n, c]$ as:

$$\mathbf{b}_2[m, n, c] = (\mathbf{w}_2 \odot \mathbf{a}[m, n, c]) + \mathbf{a}[m, n, c] \tag{7}$$

Finally, the two recalibrated feature tensors $\mathbf{b}_1[m, n, c]$ and $\mathbf{b}_2[m, n, c]$ are concatenated along the channel dimension and fed into a point-wise convolution operation using 1 filter to generate the estimated high-quality deblocked image $\hat{x}[m, n]$.

It should be pointed out that the proposed feature recalibration operation has been placed in different places of the deep network of Fig. 3.4, including the deep feature extraction stage. However, it was found that the best trade-off between the network performance and complexity is achieved when our feature recalibration operation is utilized in the high-quality image reconstruction stage.

Each of the deep networks employed in the JPEG image deblocking module is trained with the pairs of the degraded decompressed images $y_q[m,n]$ and their ground truth versions $x[m,n]$ using the Charbonnier loss function as:

$$\begin{aligned} \mathcal{L}(\mathbf{\Theta}_{g_i}) = \min_{g_i} \sqrt{\Big\| g_i(y_q[m,n]) - x[m,n] \Big\|_2^2 + \epsilon^2} \\ i = 1, ..., I \end{aligned} \tag{8}$$

where $g_i(.)$ denotes the i-th deep network used in the JPEG image deblocking module and is trained with the in-distribution QF value q_i from the set $q = \{q_1, q_2, ..., q_I\}$. Also, the value of ϵ is empirically set to 10^{-9}. The algorithm for OODNet is summarized below-

Algorithm 1 Learning Algorithm of OODNet

```
Input: Degraded decompressed images $y_q[m,n]$'s and their ground truth versions $x[m,n]$'s from the training dataset.
Output: Parameters of the four modules of the proposed OODNet.
Train the networks $f(.)$ and $o(.)$ (image QF detection and out-of-distribution detection modules) using the loss function of (2) and images $y_q[m,n]$'s, with in-distribution QF values chosen from $q = \{q_1, q_2, ..., q_M\}$.
Obtain the value of $\alpha$ based on the training images $y_q[m,n]$'s that are obtained by various in-distribution and out-of-distribution QF values.
for $i = 1, ..., I$ do
    Train the deep JPEG image deblocking networks $g_i(.)$ using the loss function of (8) and pairs of images $(y_q[m,n], x[m,n])$, with the $i$-th in-distribution QF value.
end for
Train the network $h(.)$ (image quality assessment module) using the loss function of (3) and images $g_j(y_q[m,n])$ ($j = 1, 2$), with the out-of-distribution QF values.
Return Parameters of the four modules of the proposed OODNet.
```

3.3 Experimental Results

In this section, the experimental results of the OODNet are shown. A detailed ablation study is done to show the proposed design's effectiveness and utility. Finally, a comparative study is done between OODNet and the state-of-the-art networks.

Different datasets were used for training and testing purposes. Firstly, the DIV2K [49] dataset was used to train the networks. DIV2K consists of 800 high-resolution images. For the Image QF detection module described in Section 3.2.1, compressed versions of the DIV2K images were used. Images were compressed using the JPEG compression algorithm for in-distribution QF values $q = \{q_1, q_2, ..., q_I\}$. MATLAB was used to carry out the JPEG compression and create unique folders created by compressing the original images with the in-distribution QF values. While training the Image QF Detection Module, the image patch size was $32X32$, and the batch size was 16. As the out-of-distribution detection module was jointly trained with the QF detection module, the same patch size and batch size were also used for training the out-of-distribution detection module.

DIV2k dataset was also used for training the image quality assessment module and deep jpeg image deblocking module. For training patch size of $64X64$ and their corresponding ground truth was used. The image patches were rotated and flipped to facilitate the training process. Two benchmark datasets, live1 [50] and classic5 [51], were used for testing.

The task of blind JPEG image deblocking is to develop a scheme that can respond to both in-distribution and out-of-distribution QF values. Most of the state-of-the-art deep JPEG image deblocking schemes [32], [31], [28] are trained and evaluated on the degraded decompressed images with QF values $q = \{10, 20, 30, 40\}$. Given this, we use the QF values $q = \{10, 20, 30, 40\}$ as the set of in-distribution QF values in all our experiments. Therefore, in the JPEG image deblocking module of the proposed OODNet, four deep networks were trained, each specified for handling the QF values of 10, 20, 30, and 40, respectively. Further, another four QF values, namely, 5, 15, 25, and 35, are considered for the out-of-distribution QF values to assess the robustness of the proposed and state-of-the-art deblocking networks. It is worth noting that no deep network exists in the JPEG image deblocking module of the proposed scheme that is trained with the out-of-distribution QF values. Given these explanations, the JPEG image deblocking performances are reported in all tables separately for the two sets of QF values: in-distribution QF values and out-of-distribution QF values.

In the JPEG image deblocking module of the proposed scheme, each deep network is trained with the images of the *DIV2K* dataset [49]. The training process of the deep JPEG image deblocking networks is carried out using the Adam optimizer with an initial learning rate of 10^{-4}. The total number of iterations in the training process is 200K. The value of the learning rate is decreased by a factor of 0.5 after each 50K iteration. The value of the threshold α employed for the confidence score estimated by the out-of-distribution detection of the proposed scheme is empirically set to 0.7, based on the training images of *DIV2K* dataset. The training process of each of the deep JPEG image deblocking networks is performed on a machine equipped with A6000 GPU.

PSNR-B [52] is a visual quality assessment metric that is specified for evaluating the performance of the task of JPEG image deblocking. Therefore, we use this metric and the conventional metrics, i.e., PSNR and SSIM [53], for all the ablation studies. However, since the performance of many state-of-the-art JPEG image deblocking schemes is only reported using PSNR and SSIM values, we use only these two metrics for the comparative experimentations.

3.3.1 Ablation Study Results

In this section, the ablation study results and the comparative study results are discussed. First, the impact of every part of the OODNet is justified through an ablation study. A modular design technique is used to develop the proposed blind JPEG image deblocking scheme. Instead of using the modular design technique, one could train a single deep JPEG image deblocking network with the images that are degraded with various QF values randomly chosen from the range $[q_{min}, q_{max}]$. In this case, the deep JPEG deblocking network can learn the information from all the possible QF values during its training process. However, we argue that this leads to increasing the training difficulty of the deep network (especially in the case that the range $[q_{min}, q_{max}]$ becomes large) and results in deteriorating the deblocking performance. To investigate this, we compare the performance of the proposed OODNet and its variant, in which none of the image QF detection, out-of-distribution detection, and image quality assessment modules is utilized, and the deep JPEG deblocking network of Fig. 3.4 is trained with various QF values that are randomly chosen from the range $[0, 40]$. We refer to this variant of the proposed scheme as *Variant 1*. Table I shows the results of the proposed OODNet and its *Variant 1* on the images of the *Classic5* and *LIVE1* datasets. The results of this table support our claim that the modular design technique is a more effective way of blindly performing the JPEG image deblocking task.

To investigate the impact of the image QF detection module on the JPEG image deblocking performance, we form *Variant 2* of the proposed OODNet, in which the image QF detection module is removed from the scheme. Consequently, since the functioning of the out-of-distribution detection module depends on the presence of the image QF detection module, the out-of-distribution module is also not utilized by the *Variant 2* of OODNet. Further, the image quality assessment module of *Variant 2* is trained to map the concatenation of the images produced by all the I deep networks of the JPEG image deblocking

Table 3.1: Impact of training a single deep network with various QF values on the deblocking performance. Performances are in terms of PSNR/SSIM/PSNR-B.

Networks	QF=10	QF=20	QF=30	QF=40
Variant 1	30.94/0.8326/29.63	33.05/0.8792/31.73	33.96/0.8966/32.65	34.45/0.9061/33.15
Proposed	31.21/0.8379/29.90	33.14/0.8805/31.83	34.11/0.8985/32.81	34.65/0.9083/33.34
Networks	QF=5	QF=15	QF=25	QF=35
Variant 1	28.14/0.7498/26.82	31.96/0.8601/30.64	33.56/0.8892/32.25	34.24/0.9022/32.94
Proposed	28.55/0.7678/26.93	32.24/0.8643/30.92	33.60/0.8891/32.30	34.41/0.9039/33.10

module to the image quality assessment index. Table 3.2 gives the performance of OODNet and its *Variant 2* on the images of the *Classic5* and *LIVE1* datasets. It is seen from this table that the use of the image QF detection module results in significantly enhancing the JPEG image deblocking performance. The image QF detection module, by estimating the QF value of the input degraded image, contributes in a more efficient way of looking for an appropriate deep network for reducing the blocking artifacts.

We now investigate the impact of the out-of-distribution detection module on the JPEG image deblocking performance. In this regard, a variant of the proposed OODNet, *Variant 3* is created, in which the out-of-distribution detection module is eliminated from the scheme. Since *Variant 3* does not discriminate between the degraded images obtained by the in-distribution and out-of-distribution QF values, **for all the input images**, based on the estimated QF values, the image quality assessment module selects one of the two JPEG image deblocking networks. The performances of the OODNet and its *Variant 3* on the images of the *Classic5* and *LIVE1* datasets are shown in Table 3.3. It is seen from the results of this table that the out-of-distribution detection module, by distinguishing the images obtained by the in-distribution QF values from those produced by the out-of-distribution QF values, brings on a remarkable JPEG image deblocking performance improvement.

Table 3.2: Impact of the image QF detection module on the deblocking performance. Performances are in terms of PSNR/SSIM/PSNR-B.

Networks	QF=10	QF=20	QF=30	QF=40
Variant 2	29.51/0.7978/28.20	32.43/0.8724/31.12	34.01/0.8979/32.70	34.57/0.9076/33.33
Proposed	31.21/0.8379/29.90	33.14/0.8805/31.83	34.11/0.8985/32.81	34.65/0.9083/33.34
Networks	QF=5	QF=15	QF=25	QF=35
Variant 2	26.76/0.6977/25.45	31.21/0.8457/29.89	33.35/0.8883/32.04	34.35/0.9036/33.08
Proposed	28.55/0.7678/26.93	32.24/0.8643/30.92	33.60/0.8891/32.30	34.41/0.9039/33.10

To scrutinize the influence of the image quality assessment module of the proposed OODNet on the deblocking performance, two different alternatives are considered. Firstly, a variant of the proposed scheme, *Variant 4*, in which the image quality assessment module

Table 3.3: Impact of the out-of-distribution detection module on the deblocking performance. Performances are in terms of PSNR/SSIM/PSNR-B.

Networks	QF=10	QF=20	QF=30	QF=40
Variant 3	30.16/0.8163/28.84	32.96/0.8792/31.65	33.96/0.8962/32.66	34.27/0.9022/32.96
Proposed	31.21/0.8379/29.90	33.14/0.8805/31.83	34.11/0.8985/32.81	34.65/0.9083/33.34
Networks	QF=5	QF=15	QF=25	QF=35
Variant 3	27.01/0.7087/25.70	31.91/0.8589/30.60	33.59/0.8899/32.29	34.16/0.9001/32.86
Proposed	28.55/0.7678/26.93	32.24/0.8643/30.92	33.60/0.8891/32.30	34.41/0.9039/33.10

is wholly removed is created. Secondly, two variants of the proposed scheme are created, in which, when the out-of-distribution detection module identifies that the input image is obtained by an out-of-distribution QF value based on the estimated QF, two deep deblocking networks are selected, and their predicted images are subject to the non-reference-based image quality assessment metrics. In this case, the image with a better quality assessment value is chosen as the output of the blind JPEG image deblocking method. Specifically, 2 non-reference-based image quality assessment metrics are employed, namely, BRISQUE [54], and NIQE [55], and hence, form 2 variants of the proposed OODNet, i.e., *Variant BRISQUE* and *Variant NIQE*. Tables IV and V, respectively, show the performances of the proposed OODNet and its *Variant 4*, as well as, *Variants BRISQUE* and *NIQE* on the images of the *Classic5* and *LIVE1* datasets. As seen from the results of these two tables, because the proposed image quality assessment module performs selecting the deblocked image with superior visual quality in a learnable manner, its inclusion for the task of blind JPEG image deblocking leads to higher performance.

It is mentioned in Section 3.2.4 that we use the feature recalibration operation of Fig. 3.4 in the architecture of the deep networks employed in the JPEG image deblocking module of the proposed scheme. To investigate the impact of this operation on enhancing the JPEG image deblocking performance, we form two variants of the proposed scheme, namely, *Variants 5* and *6*, in which in the first one, the feature recalibration operation is replaced by the convolution operation similar to the SwinIR network. In the second one,

Table 3.4: Impact of removing the image quality assessment module on the deblocking performance. Performances are in terms of PSNR/SSIM/PSNR-B.

Networks	QF=10	QF=20	QF=30	QF=40
Variant 4	30.21/0.8149/28.90	32.75/0.8762/31.44	33.90/0.8959/32.59	34.32/0.9035/33.01
Proposed	31.21/0.8379/29.90	33.14/0.8805/31.83	34.11/0.8985/32.81	34.65/0.9083/33.34
Networks	QF=5	QF=15	QF=25	QF=35
Variant 4	27.20/0.7152/25.89	31.71/0.8542/30.40	33.43/0.8884/32.12	34.17/0.9007/32.87
Proposed	28.55/0.7678/26.93	32.24/0.8643/30.92	33.60/0.8891/32.30	34.41/0.9039/33.10

Table 3.5: Impact of using the non-reference-based image quality assessment metrics on the deblocking performance. Performances are in terms of PSNR/SSIM/PSNR-B.

Networks	QF=10	QF=20	QF=30	QF=40
Variant BRISQUE	30.21/0.8176/28.90	32.62/0.8749/31.31	34.05/0.8982/32.74	34.58/0.9071/33.27
Variant NIQE	30.63/0.8240/29.31	32.91/0.8778/31.59	34.06/0.8981/32.75	34.62/0.9079/33.31
Proposed	31.21/0.8379/29.90	33.14/0.8805/31.83	34.11/0.8985/32.81	34.65/0.9083/33.34
Networks	QF=5	QF=15	QF=25	QF=35
Variant BRISQUE	27.30/0.7191/25.99	31.42/0.8502/30.11	33.47/0.8887/32.16	34.39/0.9038/33.09
Variant NIQE	27.44/0.7246/26.12	32.12/0.8611/30.81	33.45/0.8886/32.14	34.29/0.9034/32.99
Proposed	28.55/0.7678/26.93	32.24/0.8643/30.92	33.60/0.8891/32.30	34.41/0.9039/33.10

the feature recalibration operation is kept in the network, but its DCT branch is eliminated. Table 3.6 shows the performances of these two variants of the proposed OODNet on the images of the *Classic5* and *LIVE1* datasets. It is seen from this table that since the proposed feature recalibration operation re-weights various features based on both the spatial and DCT interactions, the performance of OODNet using our feature recalibration operation is superior to those of its *Variants 5* and *6*.

Table 3.6: Impact of the proposed feature recalibration operation on the deblocking performance. Performances are in terms of PSNR/SSIM/PSNR-B.

Networks	QF=10	QF=20	QF=30	QF=40
Variant 5	31.17/0.8365/29.85	33.02/0.8786/31.71	34.01/0.8972/32.71	34.47/0.9050/33.17
Variant 6	31.07/0.8372/29.75	33.05/0.8794/31.74	33.99/0.8969/32.68	34.47/0.9052/33.17
Proposed	31.21/0.8379/29.90	33.14/0.8805/31.83	34.11/0.8985/32.81	34.65/0.9083/33.34
Networks	QF=5	QF=15	QF=25	QF=35
Variant 5	28.13/0.7528/26.82	32.19/0.8631/30.88	33.49/0.8881/32.19	34.29/0.9022/32.99
Variant 6	28.07/0.7525/26.76	32.19/0.8635/30.87	33.49/0.8883/32.19	34.28/0.9020/32.98
Proposed	28.55/0.7678/26.93	32.24/0.8643/30.92	33.60/0.8891/32.30	34.41/0.9039/33.10

3.3.2 Comparative Study results

In this section, we compare the performance of the proposed scheme with those of the state-of-the-art deep learning-based blind and non-blind JPEG image deblocking methods on the images of the two benchmark datasets. For the comparison, we have considered only the deep learning methods. The reason behind not comparing with traditional JPEG image deblocking methods is that one of the earliest deep learning-based deblocking networks, ARCNN [4], has shown that it surpasses the performance of the best performing classical method, which is SA-DCT [2]. So, if any deblocking method performs better than ARCNN, we can confidently conclude that it will outperform the existing traditional JPEG deblocking techniques. Therefore, we compare the performance of the proposed OODNet with those of artifact reduction convolutional neural network (ARCNN) [4], trainable nonlinear reaction-diffusion (TNRD) [22], deep boosting network (DDFN-x3W) [27], residual non-local attention network (RNAN) [44], blind inception-based artifact reduction convolutional neural network (IACNN) [29], dual pixel wavelet domain with soft decoding network (DPW-SDNet) [30], deep dual-domain semi-blind network (D3SN) [28], resource-efficient blind quality enhancement (RBQE) [31], image restoration with deep denoiser prior (DRUNet) [56], residual dense network (RDN) [44] and image restoration with Swin transformer (SwinIR) [32] in Table 3.7, for the in-distribution QF values. It is to be noted that since we did not find official implementations of the two methods, D3SN and DPW-SDNet, on the Internet, we have re-implemented them based on the descriptions their authors explained in the papers. We have used the official implementations released by their authors for the other methods. It is seen from the results of this table that the proposed scheme can provide the best values in 12 out of 16 cases of the PSNR and SSIM metrics. Further, the proposed method provides 1 second best PSNR and SSIM values on the images of the two benchmark datasets. In Table. 3.7 and Table. 3.8, red and blue color denote the best and the second best performances respectivley.

Table 3.7: Comparison between the performance of various state-of-the-art schemes for the task of JPEG image deblocking, in the case of in-distribution QF values.

Dataset	Settings	ARCNN [4]		TNRD [22]		DDFN-x3W [27]		RDN [44]		RNAN [57]		IACNN [29]		DPW-SDNet [30]		D3SN [28]		RBQE [31]		DRUNet [56]		SwinIR [32]		OODNet(Proposed)	
Classic5	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	10	29.03	0.7929	29.28	0.7992	29.55	0.8086	30.00	0.8188	29.96	0.8178	29.43	0.8070	29.74	0.8124	30.09	0.8211	29.99	0.8064	30.16	0.8234	30.27	0.8249	31.21	0.8380
	20	31.15	0.8517	31.47	0.8576	31.70	0.8636	32.15	0.8699	32.11	0.8693	31.64	0.8628	31.95	0.8663	32.22	0.8710	31.88	0.8583	32.39	0.8734	32.52	0.8748	33.14	0.8806
	30	32.51	0.8806	32.78	0.8837	33.03	0.8881	33.43	0.8930	33.38	0.8924	32.93	0.8874	33.22	0.8903	33.49	0.8935	33.18	0.8864	33.59	0.8949	33.73	0.8961	34.11	0.8986
	40	33.34	0.8953	N/A	N/A	33.90	0.9023	34.27	0.9061	34.27	0.9061	33.79	0.9014	34.07	0.9039	N/A	N/A	33.99	0.9009	34.41	0.9075	34.52	0.9082	34.65	0.9083
LIVE1	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	10	27.77	0.7905	29.15	0.8111	29.39	0.8186	29.67	0.8247	29.63	0.8239	29.34	0.8199	29.53	0.8210	29.76	0.8261	29.91	0.8179	29.79	0.8278	29.86	0.8287	30.80	0.8389
	20	30.07	0.8683	31.46	0.8769	31.76	0.8839	32.07	0.8882	32.03	0.8877	31.73	0.8848	31.90	0.8854	32.09	0.8889	31.83	0.8733	32.17	0.8899	32.25	0.8909	32.81	0.8902
	30	31.41	0.9000	32.84	0.9059	33.19	0.9117	33.51	0.9153	33.45	0.9149	33.19	0.9132	33.31	0.9130	33.51	0.9154	33.11	0.9019	33.59	0.9166	33.69	0.9174	33.89	0.9121
	40	32.35	0.9173	N/A	N/A	34.20	0.9273	34.51	0.9302	34.47	0.9299	34.18	0.9283	34.30	0.9282	N/A	N/A	33.94	0.9169	34.58	0.9312	34.67	0.9317	34.54	0.9234

In Table 3.8, we now compare the performance of the proposed scheme with that of SwinIR (the second best performing scheme of Table 3.7), as well as, with the performances of the two high-performance blind JPEG image deblocking methods, D3SN [28] and RBQE [31], on the images of the two benchmark datasets, for the out-of-distribution QF values. It is seen from the results of this table that the proposed scheme can significantly outperform the other best-performing state-of-the-art deblocking methods when the input images are compressed with QF values other than those used in the network training process. In summary, based on the results given in Tables 3.7 and 3.8, it can be concluded that thanks to the use of efficient modules in the design of the proposed scheme, it can provide a superior JPEG image deblocking performance in the cases of both in-distribution and out-of-distribution QF values.

Table 3.8: Comparison between the performance of the best performing JPEG image deblocking schemes, in the case of the out-of-distribution QF values.

Dataset	Settings	D3SN [28]		RBQE [31]		SwinIR [32]		OODNet	
Classic5	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	5	28.31	0.7520	27.53	0.7294	28.05	0.7485	28.55	0.7678
	15	31.42	0.8412	31.03	0.8380	32.08	0.8572	32.24	0.8643
	25	32.82	0.8767	32.65	0.8762	32.84	0.8779	33.60	0.8891
	35	33.62	0.8957	33.62	0.8950	34.14	0.9003	34.41	0.9039
LIVE1	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	5	27.24	0.7323	27.38	0.7375	26.67	0.7077	27.62	0.7444
	15	31.27	0.8521	30.98	0.8520	31.94	0.8729	31.98	0.8717
	25	32.75	0.8918	32.58	0.8915	33.15	0.9019	33.33	0.9013
	35	33.62	0.9124	33.57	0.9110	34.17	0.9155	34.25	0.9184

3.3.3 Qualitative Performance and Comparison

To further illustrate the performance of OODNet, the qualitative performance is discussed in this section. The outputs of OODNet are compared with the state-of-the-art deblocking networks such as ARCNN [4], IACNN [29], D3SN [28], DPW-SDNet [30], RBQE [31] and DRUNet [56].

We can see from Fig. 3.5, Fig. 3.6 and Fig. 3.7 the performance of OODNet with the state-of-the-art networks mentioned above. Each of the images has been compressed with a QF value of 15. We can see from the figures that OODNet outperforms the other state-of-the-art network by producing sharper and clearer images. For example, it is seen from Fig. 3.6 that the proposed scheme better restores the structures of the bike's spring.

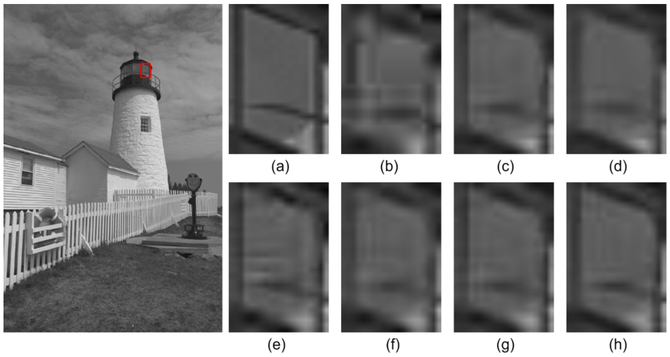


Figure 3.5: Visual quality of Paintedhouse image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (b) Decopressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) Proposed scheme.

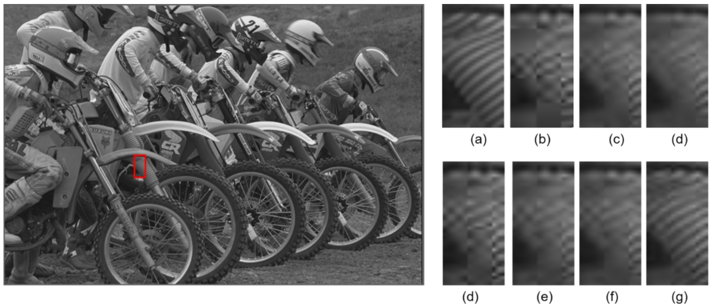


Figure 3.6: Visual quality of Bikes image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (b) Decopressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) Proposed scheme

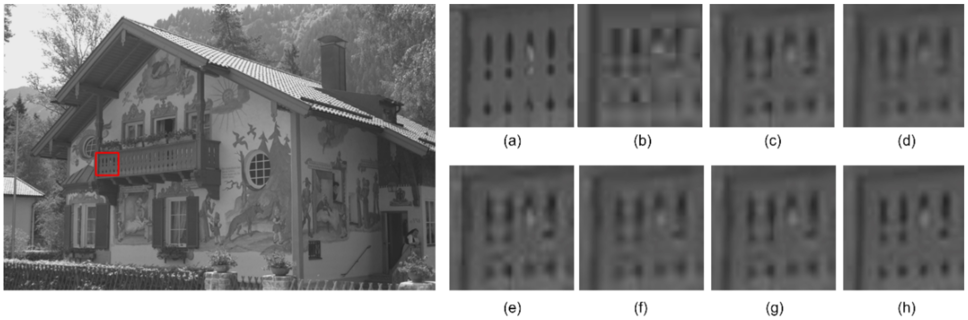


Figure 3.7: Visual quality of Lighthouse3 image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (b) Decopressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) Proposed scheme

3.4 Summary

Several state-of-the-art JPEG deblocking networks have been developed for dealing with images compressed by QF values known to the network. If the images are compressed with any other value of QF unknown to the network, the performance deteriorates significantly. In this chapter, OODNet has been introduced to effectively deal with images compressed with QF values unknown to the network which we call out-of-distribution QF values. OODNet has four separate modules to deal with the out-of-distribution QF values effectively. The utility of each of the modules has been established by conducting multiple ablation studies. A detailed comparative study has also been conducted to show the superiority of OODNet in performance with state-of-the-art networks. The visual quality diagrams also dictate that the proposed method generates images with sharper features compared to the existing state-of-the-art method.

Chapter 4

JPEG Image Deblocking using Meta-Learning

4.1 Introduction

The performance and image deblocking quality of OODNet mentioned in Chapter 3 are significantly better compared to the state-of-the-art methods. The main drawback of using OODNet is the complexity of the network. To train OODNet, four different modules: QF detection module, out-of-distribution detection module, image quality assessment module and deep deblocking module have to be trained separately. Each of the deep deblocking networks in the deep deblocking module also requires separate training. This creates a challenge because these training processes are time-consuming and take a lot of computational resources. The execution time for generating a high-quality image from a decompressed image with OODNet is higher because it uses four modules. The memory consumption for OODNet is higher as multiple models of a deep JPEG deblocking network are used in the deep deblocking module. In short, for OODNet, the improved performance is achieved at the expense of higher computational complexity, training cost and memory usage. The objective of this chapter is to introduce a deep learning scheme called MetaNet [58], which perform the task of JPEG image deblocking with reduced time complexity, training cost and memory usage. The concept of meta-learning is to make deep models generalize better to a wide variety of tasks. Usually, a deep learning network is trained on a specific task or subsets of tasks. Meta-learning aims to adapt a deep learning network to various tasks and increase its capacity and performance on tasks the network has not seen during training. This chapter is organized as follows: In Section 4.2, the proposed method is discussed; in Section 4.3, the experimental results are discussed, in Section 4.4, a comparison is done with OODNet and MetaNet and finally the concluding remarks are drawn in section 4.5.

4.2 Proposed Method

In this chapter, we propose the use of meta-learning in the task of JPEG image deblocking. A single deep JPEG image deblocking network is trained using the meta-learning algorithm. The deep deblocking network used for MetaNet is SwinIR [32]. The original network of SwinIR is used for this purpose. This is one of the main differences from OODNet of Chapter 3, where a modified version of the SwinIR network is proposed. The meta-train and meta-tests are selected carefully to facilitate the training process. In Section 4.2.1, the motivation behind using meta-learning for the task of JPEG deblocking is described, and in Section 4.2.2, the architecture of MetaNet is discussed.

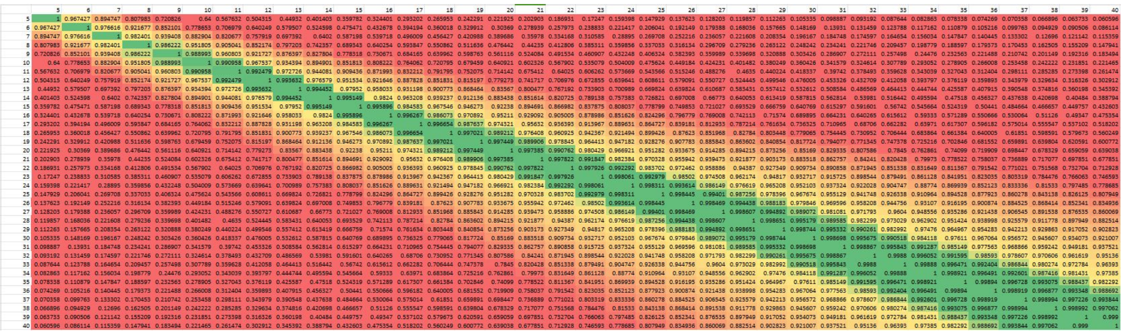
4.2.1 Motivation Behind Using Meta-Learning for JPEG Image Deblocking

Meta-learning helps a network to learn domains that closely relate to the original domain the network was originally trained with. This results in faster network training than conventional training methods, as the meta-learning algorithm focuses on increasing the capacity of a network by training the network with data very close to the test data it will later work on.

JPEG compression depends on the quantization matrix used in the quantization stage. The quantization matrix directly depends on the quality factor or QF value. In conventional networks, the network is trained for a set of QF values say, q_1, q_2, q_3 and q_M. The network performance deteriorates significantly when a QF value outside the training set is used to compress the input image. In meta-learning, two training sets are used: meta-train test and meta-test sets. The QF value plays a significant role in determining the optimum meta-test sets for the proposed MetaNet. To select the optimum range of QF values for the meta-test set, the SSIM [53] metrics can be used as a guiding tool. In figure 4.1, it can be seen how closely each of the quantization matrices is related. In the diagonal, each of the quantization matrices is compared with themselves. Hence, the value at the diagonal is always 1, which indicates maximum similarity. This unique property of the quantization matrices makes the use of meta-learning a viable option in the task of deep JPEG image deblocking. The similarity of the QF matrices can help in selecting the proper meta-test sets for the corresponding meta-train sets. If meta-train sets are considered q_1, q_2 and q_M and their corresponding meta-test sets are considered to be $q_{1(j)}, q_{2(j)}$ and $q_{I(j)}$. The SSIM matrix can help to select the optimum meta-test set $q_{I(j)}$ for its corresponding meta-train set q_I.

4.2.2 Network Architecture

The deep JPEG Image deblocking network used for MetaNet is the same as SwinIR [32] shown in Fig. 2.4. The SwinIR network has three parts: The shallow feature extraction part,



	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
5	1	0.967427	0.894747	0.807983	0.720826	0.64	0.567632	0.504315	0.44932	0.401403	0.359782	0.324401	0.293202	0.265953	0.242291	0.221925	0.202903	0.186931	0.17247	0.159398	0.147929	0.137623	0.128203	0.119857	0.112263	0.105335	0.098887	0.093192	0.087644	0.082863	0.078338	0.074269	0.070358	0.066896	0.063733	0.060596
6	0.967427	1	0.976616	0.921677	0.852101	0.778653	0.706979	0.640249	0.579507	0.524598	0.475471	0.432678	0.394194	0.360018	0.329912	0.30369	0.278939	0.257973	0.238833	0.221417	0.206041	0.192149	0.179388	0.168036	0.157665	0.148169	0.13931	0.131459	0.123788	0.117162	0.110879	0.105216	0.099763	0.094929	0.090506	0.086114
7	0.894747	0.976616	1	0.982401	0.939408	0.882904	0.820677	0.757919	0.697392	0.6402	0.587198	0.539718	0.496009	0.456427	0.420988	0.389686	0.35978	0.334168	0.310585	0.28895	0.269708	0.252216	0.236057	0.221608	0.208354	0.196167	0.184748	0.174597	0.164654	0.156034	0.147847	0.140445	0.133302	0.12696	0.121142	0.115359
8	0.807983	0.921677	0.982401	1	0.986222	0.951805	0.905041	0.852174	0.797203	0.742357	0.689343	0.640254	0.593847	0.550862	0.511636	0.476442	0.44235	0.412806	0.385311	0.359856	0.337033	0.316134	0.296709	0.279236	0.263122	0.248242	0.234241	0.221746	0.209457	0.198779	0.188597	0.179373	0.170453	0.162505	0.155209	0.147941
9	0.720826	0.852101	0.939408	0.986222	1	0.988993	0.960803	0.921727	0.876397	0.827804	0.778318	0.730671	0.684165	0.639962	0.598763	0.561116	0.524084	0.491534	0.460907	0.432248	0.406324	0.382393	0.359989	0.339698	0.320888	0.303426	0.286907	0.272111	0.257498	0.24476	0.232563	0.221488	0.210742	0.201149	0.192316	0.183494
10	0.64	0.778653	0.882904	0.951805	0.988993	1	0.990958	0.967537	0.934394	0.894901	0.851813	0.808222	0.764062	0.720795	0.679459	0.640921	0.602326	0.567902	0.535079	0.504009	0.475624	0.449184	0.424231	0.401482	0.380249	0.360426	0.341579	0.324614	0.307789	0.293052	0.278905	0.266008	0.253458	0.242222	0.231851	0.221465
11	0.567632	0.706979	0.820677	0.905041	0.960803	0.990958	1	0.992479	0.972726	0.944081	0.909436	0.871993	0.832212	0.791795	0.752075	0.714142	0.675412	0.64025	0.606262	0.573669	0.543566	0.515246	0.488276	0.4635	0.440224	0.418337	0.39742	0.378493	0.359628	0.343039	0.327043	0.312404	0.298111	0.285285	0.273398	0.261474
12	0.504315	0.640249	0.757919	0.852174	0.921727	0.967537	0.992479	1	0.993632	0.976579	0.951534	0.921646	0.887828	0.851831	0.815197	0.779273	0.741717	0.706976	0.672855	0.639641	0.608611	0.579091	0.550727	0.524445	0.499546	0.476005	0.453326	0.432709	0.412058	0.393797	0.376119	0.359893	0.343979	0.329634	0.316326	0.302912
13	0.44932	0.579507	0.697392	0.797203	0.876397	0.934394	0.972726	0.993632	1	0.994452	0.97952	0.958033	0.931198	0.900773	0.868464	0.83567	0.800477	0.767192	0.733903	0.700989	0.669824	0.639824	0.610687	0.583431	0.557412	0.532612	0.508584	0.486569	0.464413	0.444744	0.425587	0.407915	0.390548	0.374816	0.360198	0.345392
14	0.401403	0.524598	0.6402	0.742357	0.827804	0.894901	0.944081	0.976579	0.994452	1	0.995149	0.9824	0.963208	0.939237	0.912136	0.883438	0.851614	0.820725	0.789138	0.757383	0.726821	0.697008	0.66773	0.640053	0.613419	0.587815	0.562814	0.53981	0.516442	0.495594	0.47518	0.456327	0.437638	0.420698	0.40484	0.388794
15	0.359782	0.475471	0.587198	0.689343	0.778318	0.851813	0.909436	0.951534	0.97952	0.995149	1	0.995896	0.984583	0.967546	0.946273	0.92238	0.894691	0.866982	0.837875	0.808037	0.778799	0.749853	0.721027	0.693529	0.666759	0.640769	0.615297	0.591601	0.56742	0.545664	0.524319	0.50441	0.484664	0.466657	0.449757	0.432603
16	0.324401	0.432678	0.539718	0.640254	0.730671	0.808222	0.871993	0.921646	0.958033	0.9824	0.995896	1	0.996267	0.986073	0.970892	0.95211	0.929092	0.905005	0.878986	0.851626	0.824296	0.796779	0.769008	0.742113	0.71574	0.689895	0.664231	0.640265	0.615612	0.59333	0.571289	0.550666	0.530064	0.51126	0.49347	0.475354
17	0.293202	0.394194	0.496009	0.593847	0.684165	0.764062	0.832212	0.887828	0.931198	0.963208	0.984583	0.996267	1	0.996654	0.987637	0.974321	0.95632	0.936393	0.913967	0.889631	0.864727	0.839181	0.812933	0.787214	0.761634	0.736325	0.710965	0.68706	0.662282	0.63971	0.617307	0.596182	0.575014	0.555547	0.537102	0.518202
18	0.265953	0.360018	0.456427	0.550862	0.639962	0.720795	0.791795	0.851831	0.900773	0.939237	0.967546	0.986073	0.996654	1	0.997021	0.989212	0.976408	0.960925	0.942367	0.921494	0.899426	0.87623	0.851968	0.82784	0.803448	0.779065	0.754445	0.730952	0.706444	0.683864	0.661384	0.640005	0.61851	0.598591	0.579673	0.560249
19	0.242291	0.329912	0.420988	0.511636	0.598763	0.679459	0.752075	0.815197	0.868464	0.912136	0.946273	0.970892	0.987637	0.997021	1	0.997449	0.989906	0.978845	0.964413	0.947182	0.928276	0.907783	0.885843	0.863602	0.840854	0.817724	0.794077	0.771345	0.747378	0.725216	0.702846	0.681552	0.659891	0.639804	0.620591	0.600772
20	0.221925	0.30369	0.389686	0.476442	0.561116	0.640921	0.714142	0.779273	0.83567	0.883438	0.92238	0.95211	0.974321	0.989212	0.997449	1	0.997385	0.990762	0.980429	0.966921	0.951282	0.933675	0.914285	0.894215	0.873256	0.85169	0.829335	0.807586	0.7845	0.762861	0.74099	0.719909	0.698447	0.678329	0.659059	0.639038
21	0.202903	0.278939	0.35978	0.44235	0.524084	0.602326	0.675412	0.741717	0.800477	0.851614	0.894691	0.929092	0.95632	0.976408	0.989906	0.997385	1	0.997822	0.991847	0.982384	0.970328	0.955942	0.939473	0.921877	0.903173	0.883518	0.862757	0.84241	0.820428	0.79973	0.778522	0.758037	0.736889	0.717077	0.697851	0.677851
22	0.186931	0.257973	0.334168	0.412806	0.491534	0.567902	0.64025	0.706976	0.767192	0.820725	0.866982	0.905005	0.936393	0.960925	0.978845	0.990762	0.997822	1	0.997926	0.992292	0.983702	0.972462	0.958886	0.94387	0.927349	0.909734	0.890858	0.871945	0.851338	0.831649	0.811367	0.791542	0.771021	0.751568	0.732704	0.712928
23	0.17247	0.238833	0.310585	0.385311	0.460907	0.535079	0.606262	0.672855	0.733903	0.789138	0.837875	0.878986	0.913967	0.942367	0.964413	0.980429	0.991847	0.997926	1	0.998061	0.992979	0.98502	0.974508	0.962174	0.94817	0.932717	0.915725	0.898544	0.879491	0.861128	0.841951	0.823035	0.803319	0.784476	0.766063	0.746593
24	0.159398	0.221417	0.28895	0.359856	0.432248	0.504009	0.573669	0.639641	0.700989	0.757383	0.808037	0.851626	0.889631	0.921494	0.947182	0.966921	0.982384	0.992292	0.998061	1	0.998311	0.993614	0.986149	0.976619	0.965208	0.952103	0.937324	0.922028	0.904747	0.88774	0.869939	0.852123	0.833336	0.81533	0.797485	0.778685
25	0.147929	0.206041	0.269708	0.337033	0.406324	0.475624	0.543566	0.608611	0.669824	0.726821	0.778799	0.824296	0.864727	0.899426	0.928276	0.951282	0.970328	0.983702	0.992979	0.998311	1	0.998445	0.99401	0.987256	0.978396	0.967674	0.955129	0.941748	0.926338	0.910964	0.894528	0.877923	0.860278	0.843138	0.826125	0.807949
26	0.137623	0.192149	0.252216	0.316134	0.382393	0.449184	0.515246	0.579091	0.639824	0.697008	0.749853	0.796779	0.839181	0.87623	0.907783	0.933675	0.955942	0.972462	0.98502	0.993614	0.998445	1	0.998469	0.994438	0.988183	0.979846	0.969596	0.958208	0.944756	0.93107	0.916195	0.900874	0.884525	0.868414	0.852341	0.834936
27	0.128203	0.179388	0.236057	0.296709	0.359989	0.424231	0.488276	0.550727	0.610687	0.66773	0.721027	0.769008	0.812933	0.851968	0.885843	0.914285	0.939473	0.958886	0.974508	0.986149	0.99401	0.998469	1	0.998607	0.994892	0.989072	0.981081	0.971793	0.9604	0.948556	0.935286	0.921438	0.906545	0.891538	0.876535	0.860069
28	0.119857	0.168036	0.221608	0.279236	0.339698	0.401482	0.4635	0.524445	0.583431	0.640053	0.693529	0.742113	0.787214	0.82784	0.863602	0.894215	0.921877	0.94387	0.962174	0.976619	0.987256	0.994438	0.998607	1	0.998651	0.995179	0.989585	0.982299	0.973029	0.962902	0.951424	0.938998	0.925579	0.911778	0.897949	0.882514
29	0.112263	0.157665	0.208354	0.263122	0.320888	0.380249	0.440224	0.499546	0.557412	0.613419	0.666759	0.71574	0.761634	0.803448	0.840854	0.873256	0.903173	0.927349	0.94817	0.965208	0.978396	0.988183	0.994892	0.998651	1	0.998744	0.995332	0.990261	0.982992	0.97476	0.964967	0.954283	0.942213	0.929863	0.917052	0.902823
30	0.105335	0.148169	0.196167	0.248242	0.303426	0.360426	0.418337	0.476005	0.532612	0.587815	0.640769	0.689895	0.736325	0.779065	0.817724	0.85169	0.883518	0.909734	0.932717	0.952103	0.967674	0.979846	0.989072	0.995179	0.998744	1	0.998698	0.995675	0.990518	0.984118	0.97611	0.967064	0.956572	0.945607	0.934073	0.921007
31	0.098887	0.13931	0.184748	0.234241	0.286907	0.341579	0.39742	0.453326	0.508584	0.562814	0.615297	0.664231	0.710965	0.754445	0.794077	0.829335	0.862757	0.890858	0.915725	0.937324	0.955129	0.969596	0.981081	0.989585	0.995332	0.998698	1	0.998867	0.995843	0.991287	0.985149	0.977563	0.968866	0.959242	0.949181	0.937521
32	0.093192	0.131459	0.174597	0.221746	0.272111	0.324614	0.378493	0.432709	0.486569	0.53981	0.591601	0.640265	0.68706	0.730952	0.771345	0.807586	0.84241	0.871945	0.898544	0.922028	0.941748	0.958208	0.971793	0.982299	0.990261	0.995675	0.998867	1	0.9988	0.996052	0.991595	0.98593	0.978607	0.970606	0.961619	0.95136
33	0.087644	0.123788	0.164654	0.209457	0.257498	0.307789	0.359628	0.412058	0.464413	0.516442	0.56742	0.615612	0.662282	0.706444	0.747378	0.7845	0.820428	0.851338	0.879491	0.904747	0.926338	0.944756	0.9604	0.973029	0.982992	0.990518	0.995843	0.9988	1	0.99888	0.996471	0.992404	0.986844	0.980274	0.972784	0.96393
34	0.082863	0.117162	0.156034	0.198779	0.24476	0.293052	0.343039	0.393797	0.444744	0.495594	0.545664	0.59333	0.63971	0.683864	0.725216	0.762861	0.79973	0.831649	0.861128	0.88774	0.910964	0.93107	0.948556	0.962902	0.97476	0.984118	0.991287	0.996052	0.99888	1	0.998921	0.996491	0.992601	0.987416	0.981431	0.97385
35	0.078338	0.110879	0.147847	0.188597	0.232563	0.278905	0.327043	0.376119	0.425587	0.47518	0.524319	0.571289	0.617307	0.661384	0.702846	0.74099	0.778522	0.811367	0.841951	0.869939	0.894528	0.916195	0.935286	0.951424	0.964967	0.97611	0.985149	0.991595	0.996471	0.998921	1	0.99894	0.996728	0.993075	0.988437	0.982292
36	0.074269	0.105216	0.140445	0.179373	0.221488	0.266008	0.312404	0.359893	0.407915	0.456327	0.50441	0.550666	0.596182	0.640005	0.681552	0.719909	0.758037	0.791542	0.823035	0.852123	0.877923	0.900874	0.921438	0.938998	0.954283	0.967064	0.977563	0.98593	0.992404	0.996491	0.99894	1	0.998919	0.996877	0.993348	0.988692
37	0.070358	0.099763	0.133302	0.170453	0.210742	0.253458	0.298111	0.343979	0.390548	0.437638	0.484664	0.530064	0.575014	0.61851	0.659891	0.698447	0.736889	0.771021	0.803319	0.833336	0.860278	0.884525	0.906545	0.925579	0.942213	0.956572	0.968866	0.978607	0.986844	0.992601	0.996728	0.998919	1	0.998994	0.997226	0.993844
38	0.066896	0.094929	0.12696	0.162505	0.201149	0.242222	0.285285	0.329634	0.374816	0.420698	0.466657	0.51126	0.555547	0.598591	0.639804	0.678329	0.717077	0.751568	0.784476	0.81533	0.843138	0.868414	0.891538	0.911778	0.929863	0.945607	0.959242	0.970606	0.980274	0.987416	0.993075	0.996877	0.998994	1	0.998992	0.997062
39	0.063733	0.090506	0.121142	0.155209	0.192316	0.231851	0.273398	0.316326	0.360198	0.40484	0.449757	0.49347	0.537102	0.579673	0.620591	0.659059	0.697851	0.732704	0.766063	0.797485	0.826125	0.852341	0.876535	0.897949	0.917052	0.934073	0.949181	0.961619	0.972784	0.981431	0.988437	0.993348	0.997226	0.998992	1	0.999
40	0.060596	0.086114	0.115359	0.147941	0.183494	0.221465	0.261474	0.302912	0.345392	0.388794	0.432603	0.475354	0.518202	0.560249	0.600772	0.639038	0.677851	0.712928	0.746593	0.778685	0.807949	0.834936	0.860069	0.882514	0.902823	0.921007	0.937521	0.95136	0.96393	0.97385	0.982292	0.988692	0.993844	0.997062	0.999	1

Figure 4.1: SSIM Matrix Showing the Similarity Between the Quantization Matrices Corresponding to the QF Values, The values are higher in the Diagonal Region of the Matrix

the deep feature extraction part, and the high-quality image reconstruction part. The difference between SwinIR and MetaNet is in the way this network is trained, which makes it perform significantly better on tasks that it has not been trained on. Meta-learning can adapt SwinIR to learn the general features better and it translates into superior performance compared to the state-of-the-art networks as shown in 4.3.2

4.2.3 Learning Algorithm

In this section, the learning algorithm of MetaNet is discussed in detail. The inspiration behind this algorithm comes from Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks or MAML [59]. MAML can be applied to any network trained with gradient descent.

Firstly, the parameter initialization is done. SwinIR models trained with QF 10, 20, 30, and 40 are used to initialize the network. Then, the average of these four weights is used to initialize the SwinIR model to find a good starting point. The effectiveness of initializing using the average weights of the mentioned networks is shown in Table. 4.3.

Then, meta-train sets are defined. For training MetaNet, the images of DIV2k [49] are compressed using Qf values of 10, 20, 30, and 40. The meta-test sets are defined using the

SSIM matrix shown in Fig. 4.1. The optimum set of meta-test sets is determined to be in the 98 percentile similarity of each corresponding meta-train set. For example, the meta train set of QF 10 corresponds to a meta test set of QF = $9, 11$.

First, a batch from the first meta-train set is passed through the main network, and the updated weights are saved in a temporary network. Then, a meta-test set batch corresponding to the meta-train set is passed through the temporary network, and the loss is calculated. The loss is saved for later use. The same is done for the rest of the three meta-train sets and their corresponding meta-test tests. Finally, the losses related to each meta test set $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3,$ and $\mathcal{L}_4$ are calculated. Then, the average of the four losses is calculated, and backpropagation is used to update the weights of the primary model. This process is continued till convergence. The loss function used for this purpose is Charbonnier loss. If the degraded decompressed images are denoted as $y_q[m, n]$ and their ground truth versions are denoted as $x[m, n]$, the Charbonnier loss function is:

$$\mathcal{L}(\mathbf{\Theta}_{g_i}) = \min_{g_i} \sqrt{\left\| g_i(y_q[m, n]) - x[m, n] \right\|_2^2 + \epsilon^2} \tag{1}$$

where $g_i(.)$ denotes the SwinIR network used for meta-learning.

The training process can be visualized in Fig. 4.2. The figure shows how the network is trained. The parameter for the main network is denoted as Θ. Meta-train sets, q_1, q_2, q_3 up to q_M are passed to the main model. Temporary models are generated by updating the parameter Θ based on the meta-train sets passed to the network. The main model parameters are unchanged at this stage. The intermediate model parameters are denoted as θ_1, θ_2, θ_3 up to θ_I. Then the meta-test sets corresponding to the meta-train sets, $q_{1(j)}$,$q_{2(j)}$,$q_{3(j)}$ up to $q_{I(j)}$ are passed through the intermediate models and the losses are calculated. Then the average of these losses is calculated, and backpropagation is used to update the weights of the

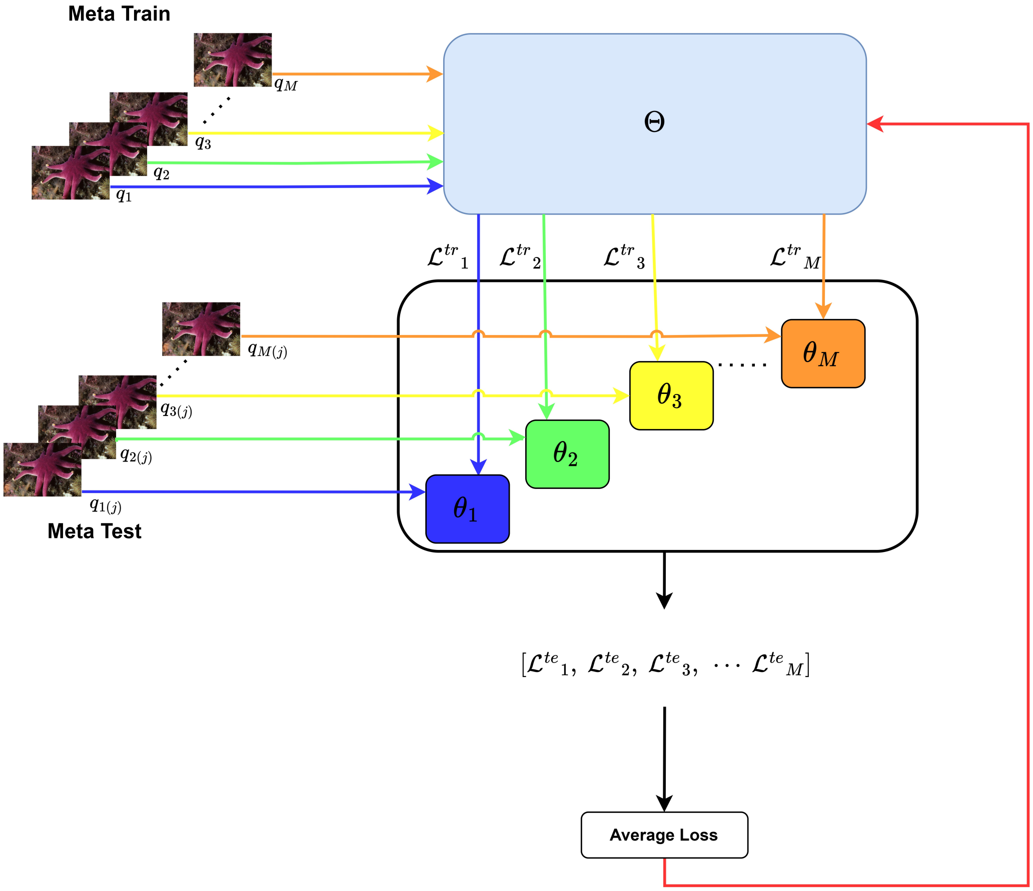


Figure 4.2: Scheme for the Metanet Training Algorithm

main model Θ. This allows the model to learn the general representation of the tasks. The whole process is repeated till the model converges. Two different learning rates are used for training the model: adaptive learning rate and meta-learning rate. The meta-learning rate or meta-test learning rate is half the value of the adaptive or meta-train learning rate. For our experiments, the adaptive learning rate was 10^{-4}, and the meta-learning rate was $5X10^{-4}$. The learning algorithm is shown below-

Algorithm 2 Learning Algorithm of MetaNet

```
Input: Meta-train sets $q_M$, the corresponding meta-test sets $q_{M(j)}$ an their corresponding ground truths.
Input: Meta-train learning rate $\alpha$, Meta-test learning rate $\beta$.
Output: Parameters of the main model $\Theta$.
Initialize the main model by averaging the weights of SwinIR model trained with QF 10, 20, 30 and 40.
Calculate the training loss $L^{tr}(\Theta)$ by passing the meta-train sets to the main model.
Calculate adapted parameters of intermediate models with gradient descent:
    $\theta_i = \Theta - \alpha \nabla_{\Theta} \mathcal{L}^{tr}$
Update $\Theta$ with respect to the average meta-test loss:
    $\Theta = \Theta - \beta \nabla_{\Theta} \frac{\sum_{i=1}^{J} L_i^{tr}(\theta_i)}{J-1}$
Repeat the steps until convergence
Return Parameters of the main model $\Theta$.
```

4.3 Experimental Results

In this section, the experimental results are given. A detailed ablation study is done to show the proposed method's effectiveness. To demonstrate the improved performance of MetaNet, a comparative analysis is done to compare its performance with that of state-of-the-art networks.

For training MetaNet, the DIV2k, [49] dataset was used. The ground truth images from the DIV2k were compressed using the QF values corresponding to the meta train and test sets to create the meta train and test tests.

4.3.1 Ablation Study

In this section, the ablation study is discussed in detail. As per the experiments, the best performance of MetaNet is achieved when the average weight initialization is used, and 98 percentile similarity is used. A total of seven ablation studies are done. The ablation study has been done on classic5 [51] dataset. Red shows the best performance in each of the tables. The details of the ablation studies are given below:

Ablation 1: Single SwinIR model trained with all Qf values: To emphasize why a meta-learning algorithm is required for optimum performance, a SwinIR model has been trained with all the QF values in the range 10 to 40. The results are shown in Table. 4.1. It can be seen that training a single model with images compressed by QF values from 10 to 40 does not produce satisfactory results and produces inferior results compared to MetaNet.

Table 4.1: Impact of Using a Single SwinIR Model Trained with QF Values Between 10 and 40

Dataset	Settings	Ablation 1		Proposed	
Classic5	QF	PSNR	SSIM	PSNR	SSIM
	5	27.32	0.7222	28.54	0.7599
	15	32.19	0.8620	32.35	0.8642
	25	33.58	0.8885	33.66	0.8907
	35	34.11	0.8991	34.37	0.9038
	10	30.49	0.8241	31.13	0.8359
	20	33.08	0.8791	33.12	0.8806
	30	33.90	0.8948	34.07	0.8982
	40	34.24	0.9018	34.61	0.9078

Ablation 2: In this ablation study, MetaNet is not trained with the meta-training sets comprising QF values 10,20,30,40. Rather, the values of QF are chosen randomly. It again results in inferior performance, as seen from Table. 4.2.

Table 4.2: Impact of SwinIR Meta Model Trained with Random QF Values for Meta Train Set

Dataset	Settings	Ablation 2		Proposed	
Classic5	QF	PSNR	SSIM	PSNR	SSIM
	5	27.28	0.7222	28.54	0.7599
	15	32.20	0.8620	32.35	0.8642
	25	33.58	0.8885	33.66	0.8907
	35	34.01	0.8991	34.37	0.9038
	10	30.48	0.8241	31.13	0.8359
	20	33.12	0.8791	33.12	0.8806
	30	33.87	0.8948	34.07	0.8982
	40	34.13	0.9018	34.61	0.9078

Ablation 3: In this ablation study, the effectiveness of initiating the model with the average weights of the SwinIR model trained with QF 10, 20, 30, and 40 are shown in Table. 4.3, it can be seen that the model initiated with the average weights of the SwinIR model trained with QF 10, 20, 30, and 40 performs better in the majority of the cases compared to models initiated with weights of SwinIR model trained with QF 10, QF 20, QF 30 and QF 40 individually.

Table 4.3: Impact of Model Weight Initialization on the Network

Ablation 3											
Dataset	Settings	Ablation 3A		Ablation 3B		Ablation 3C		Ablation 3D		Proposed	
Classic5	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	5	28.04	0.7482	27.18	0.7160	27.02	0.7079	26.97	0.7060	28.54	0.7599
	15	32.13	0.8590	32.17	0.8626	31.78	0.8568	31.60	0.8537	32.35	0.8642
	25	32.86	0.8758	33.62	0.8889	33.66	0.8907	33.57	0.8901	33.66	0.8907
	35	33.11	0.8830	34.03	0.8968	34.32	0.9020	34.38	0.9035	34.37	0.9038
	10	31.17	0.8365	30.36	0.8220	30.01	0.8133	29.88	0.8100	31.13	0.8359
	20	32.60	0.8701	33.16	0.8808	32.94	0.8789	32.77	0.8767	33.12	0.8806
	30	33.00	0.8796	33.91	0.8946	34.09	0.8977	34.08	0.8980	34.07	0.8982
	40	33.23	0.8859	34.14	0.8992	34.46	0.9042	34.57	0.9064	34.61	0.9078

Ablation 4: In this ablation study, an experiment is done to show the effectiveness of the particular similarity threshold utilized in MetaNet. As discussed in the earlier section, the proposed network utilizes meta-test sets that fall in the SSIM similarity range of the 98 percentile. For ablation 4, the SSIM similarity range is set to the 94 percentile of the meta-train sets. The results are shown in Table. 4.4.

Ablation 5: In this ablation study, an experiment is done to show the effectiveness of the particular similarity threshold utilized in MetaNet. As discussed in the earlier section, the proposed network utilizes meta-test sets that fall in the SSIM similarity range of the 98 percentile. For ablation 5, the SSIM similarity range is set to the 96 percentile of the meta-train sets. The results are shown in Table. 4.5.

Table 4.4: Impact of Training the Meta Model With Meta Test Sets Within 94 Percentile Similarity with Corresponding Meta Train Sets

Ablation 4					
Dataset	Settings	Ablation 4		Proposed	
Classic5	QF	PSNR	SSIM	PSNR	SSIM
	5	27.43	0.7274	28.54	0.7599
	15	32.24	0.8638	32.35	0.8642
	25	33.60	0.8893	33.66	0.8907
	35	34.15	0.9003	34.37	0.9038
	10	30.61	0.8282	31.13	0.8359
	20	33.13	0.8803	33.12	0.8806
	30	33.94	0.8956	34.07	0.8982
	40	34.29	0.9030	34.61	0.9078

Table 4.5: Impact of Training the Meta Model With Meta Test Sets Within 96 Percentile Similarity with Corresponding Meta Train Sets

Ablation 5					
Dataset	Settings	Ablation 5		Proposed	
Classic5	QF	PSNR	SSIM	PSNR	SSIM
	5	27.30	0.7203	28.54	0.7599
	15	32.20	0.8634	32.35	0.8642
	25	33.64	0.8897	33.66	0.8907
	35	34.15	0.9001	34.37	0.9038
	10	30.48	0.8249	31.13	0.8359
	20	33.12	0.8803	33.12	0.8806
	30	33.97	0.8959	34.07	0.8982
	40	34.30	0.9028	34.61	0.9078

Ablation 6: In this ablation study, an experiment is done to show the utility of the similarity threshold utilized in MetaNet. The proposed network utilizes meta-test sets that fall in the SSIM similarity range of 98 percentile. In this ablation study, no threshold is used. Meta-test sets are randomly selected from the set of images compressed with QF values in the range of 1 to 40. As seen from Table. 4.6, removing the threshold deteriorates the performance compared to the proposed method.

Table 4.6: Impact of Training the Meta Model With Randomly Selected Meta Test Sets

Ablation 6					
Dataset	Settings	Ablation 6		Proposed	
	QF	PSNR	SSIM	PSNR	SSIM
Classic5	5	27.29	0.7212	28.54	0.7599
	15	32.13	0.8623	32.35	0.8642
	25	33.65	0.8897	33.66	0.8907
	35	34.21	0.9004	34.37	0.9038
	10	30.42	0.8239	31.13	0.8359
	20	33.12	0.8801	33.12	0.8806
	30	34.00	0.8961	34.07	0.8982
	40	34.34	0.9027	34.61	0.9078

Ablation 7: The final ablation study is done by using the same sets for the meta-train set and meta-test sets. For example, if the meta train set has images of QF value 10, the corresponding meta test set also has images of QF value 10. This ablation study can also be interpreted as using meta-train sets that fall in the SSIM similarity range of 100 percentile. The results are shown in Table. 4.7.

Table 4.7: Impact of Training the Meta Model With Meta-test sets Similar to their corresponding meta-train sets

Ablation 7					
Dataset	Settings	Ablation 7		Proposed	
	QF	PSNR	SSIM	PSNR	SSIM
Classic5	5	27.26	0.7141	28.54	0.7599
	15	32.19	0.8622	32.35	0.8642
	25	33.65	0.8894	33.66	0.8907
	35	34.20	0.9003	34.37	0.9038
	10	30.49	0.8239	31.13	0.8359
	20	33.13	0.8798	33.12	0.8806
	30	33.99	0.8957	34.07	0.8982
	40	34.33	0.9028	34.61	0.9078

4.3.2 Comparative Study

In this section, we compare the performance of the proposed scheme with those of the state-of-the-art deep learning-based blind and non-blind JPEG image deblocking methods on the images of the two benchmark datasets, in the case of the in-distribution and out-of-distribution QF values. For the comparison, we have considered only the deep learning methods. The reason behind not comparing with traditional JPEG image deblocking methods is that one of the earliest deep learning-based deblocking networks, ARCNN [4], has shown that it surpasses the performance of the best performing classical method, which is SA-DCT [2]. So, if any deblocking method performs better than ARCNN, we can confidently conclude that it will outperform the traditional JPEG deblocking techniques. Therefore, we compare the performance of the proposed MetaNet with those of artifact reduction convolutional neural network (ARCNN) [4], trainable nonlinear reaction-diffusion (TNRD) [22], deep boosting network (DDFN-x3W) [27], residual non-local attention network (RNAN) [44], blind inception-based artifact reduction convolutional neural network (IACNN) [29], dual pixel wavelet domain with soft decoding network (DPW-SDNet) [30], deep dual-domain semi-blind network (D3SN) [28], resource-efficient blind quality enhancement (RBQE) [31], image restoration with deep denoiser prior (DRUNet) [56], residual dense network (RDN) [44] and image restoration with Swin transformer (SwinIR) [32]. In Table.4.8 comparison is done with the state-of-the-art deep deblocking networks on QF 10, 20, 30 and 40. We can call these QF values conventional values as most state-of-the-art networks are trained on these settings.

In Table.4.9, comparison is done with the state-of-the-art deep deblocking networks on QF 5, 15, 25 and 35. We can call these QF values unconventional QF values as most state-of-the-art networks are not trained on these settings.

Table 4.8: Comparison between the performance of various state-of-the-art schemes for the task of JPEG image deblocking, in the case of conventional QF values.

Dataset	Settings	ARCNN [4]		TNRD [22]		DDFN-x3W [27]		RNAN [57]		IACNN [29]		DPW-SDNet [30]		D3SN [28]		RBQE [31]		DRUNet [56]		RDN [44]		SwinIR [43]		MetaNet(Proposed)	
Classic5	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	10	29.03	0.7929	29.28	0.7992	29.55	0.8086	29.96	0.8178	29.43	0.8070	29.74	0.8124	30.09	0.8211	29.99	0.8064	30.16	0.8234	30.00	0.8188	30.27	0.8249	31.13	0.8359
	20	31.15	0.8517	31.47	0.8576	31.70	0.8636	32.11	0.8693	31.64	0.8628	31.95	0.8663	32.22	0.8711	31.88	0.8583	32.39	0.8734	32.15	0.8699	32.52	0.8748	33.12	0.8806
	30	32.51	0.8806	32.78	0.8837	33.03	0.8881	33.38	0.8924	32.93	0.8874	33.22	0.8903	33.49	0.8935	33.18	0.8864	33.59	0.8949	33.43	0.8930	33.73	0.8961	34.07	0.8982
	40	33.34	0.8953	NA	NA	33.90	0.9023	34.27	0.9061	33.79	0.9014	34.07	0.9039	NA	NA	33.99	0.9009	34.41	0.9075	34.27	0.9061	34.52	0.9082	34.61	0.9078
Live1	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	10	27.77	0.7905	29.15	0.8111	29.39	0.8186	29.63	0.8239	29.34	0.8199	29.53	0.8210	29.76	0.8261	29.91	0.8179	29.79	0.8278	29.67	0.8247	29.86	0.8287	30.74	0.8374
	20	30.07	0.8683	31.46	0.8769	31.76	0.8839	32.03	0.8877	31.73	0.8848	31.90	0.8854	32.09	0.8889	31.83	0.8733	32.17	0.8899	32.07	0.8882	32.25	0.8909	32.80	0.8903
	30	31.41	0.9000	32.84	0.9059	33.19	0.9117	33.45	0.9149	33.19	0.9132	33.31	0.9130	33.51	0.9154	33.11	0.9019	33.59	0.9166	33.51	0.9153	33.69	0.9174	33.87	0.9120
	40	32.35	0.9173	NA	NA	34.20	0.9273	34.47	0.9299	34.18	0.9283	34.30	0.9282	NA	NA	33.94	0.9169	34.58	0.9312	34.51	0.9302	34.67	0.9317	34.50	0.9231

Table 4.9: Comparison between the performance of the best performing JPEG image deblocking schemes, in the case of the out-of-distribution QF values.

Dataset	Settings	D3SN [28]		RBQE [31]		SwinIR [32]		MetaNet	
Classic5	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	5	28.31	0.7520	27.53	0.7294	28.05	0.7485	28.54	0.7599
	15	31.42	0.8412	31.03	0.8380	32.08	0.8572	32.35	0.8642
	25	32.82	0.8767	32.65	0.8762	32.84	0.8779	33.66	0.8907
	35	33.62	0.8957	33.62	0.8950	34.14	0.9003	34.37	0.9038
Live1	QF	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
	5	27.24	0.7323	27.38	0.7375	26.67	0.7077	28.11	0.7573
	15	31.27	0.8521	30.98	0.8520	31.94	0.8729	31.99	0.8713
	25	32.75	0.8918	32.58	0.8915	33.15	0.9019	33.39	0.9030
	35	33.62	0.9124	33.57	0.9110	34.17	0.9155	34.24	0.9186

4.3.3 Qualitative Performance and Comparison

To illustrate the performance of MetaNet further, the qualitative performance is discussed in this section. The outputs of MetaNet are compared with the state-of-the-art deblocking networks such as ARCNN [4], IACNN [29], D3SN [28], DPW-SDNet [30], RBQE [31] and DRUNet [56].

It can be seen from Fig. 4.3, Fig. 4.4, Fig. 4.5 and Fig. 4.6 that the proposed MetaNet produced sharper images compared to the other state-of-the-art networks. MetaNet also is able to restore the image better compared to the other state-of-the-art networks.

4.4 Comparison between OODNet and MetaNet

In this section, a comparative study is done with the network proposed in Chapter. 3, OODNet and the network proposed in this chapter, MetaNet.

A comparison is done between OODNet and MetaNet in Table. 4.10. The comparison is done on Live1 and classic5 datasets. In the table, red color denotes the best performance. It can be seen from Table. 4.10 that OODNet performs better than MetaNet in most cases,

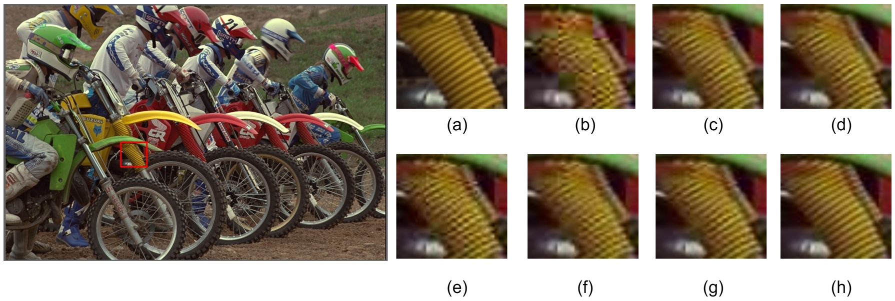


Figure 4.3: Visual quality of Bikes image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (a) Ground truth. (b) Decompressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) MetaNet.

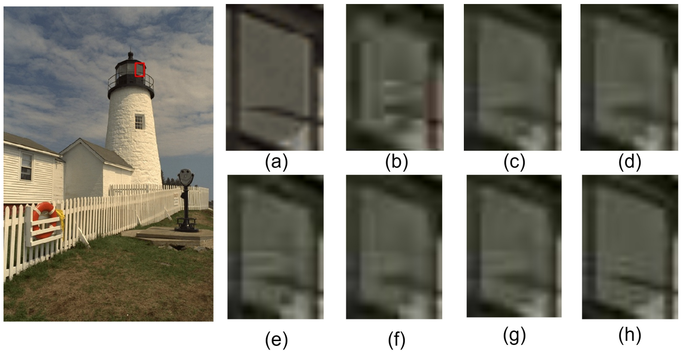


Figure 4.4: Visual quality of Lighthouse3 image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (b) Decompressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) MetaNet.

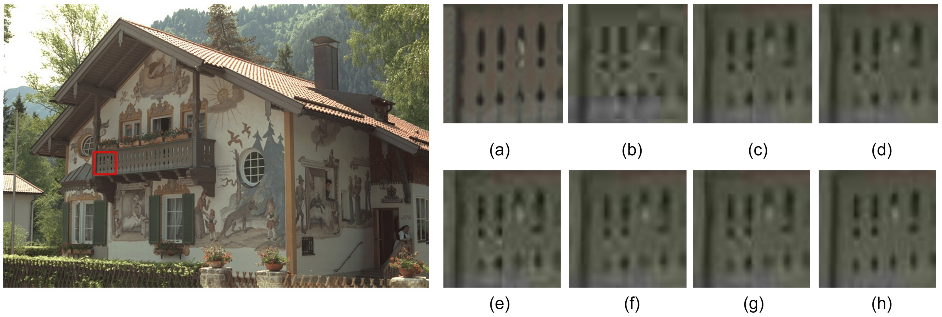


Figure 4.5: Visual quality of paintedhouse image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (b) Decompressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) MetaNet.

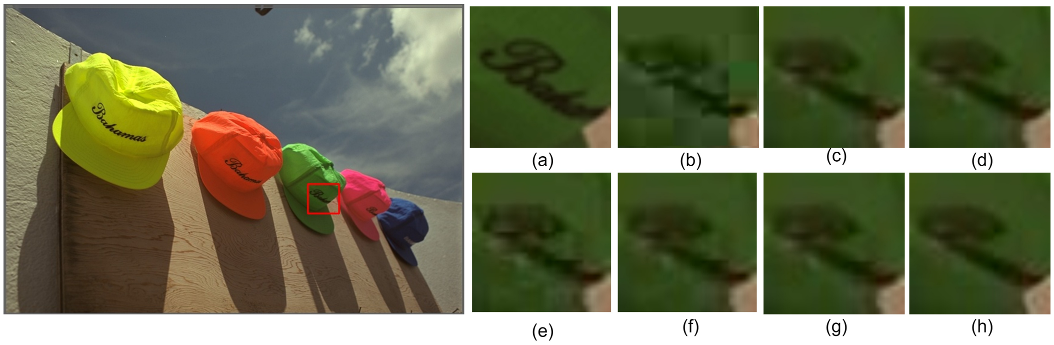


Figure 4.6: Visual quality of hats image from LIVE1 dataset degraded by QF q = 15 and restored by various state-of-the-art schemes. (a) Ground truth. (b) Decompressed Input Image. (c) IACNN. (d) D3SN. (e) DPW-SDNet. (f) RBQE. (g) DRUNet. (h) MetaNet.

which is obvious as OODNet is more complicated than MetaNet. What also can be seen from the table is the difference between psnr and ssim values in the case of MetaNet and OODNet, which are very low. In Fig. 4.7, it is seen that the runtime complexity for MetaNet is the same as SwinIR [32] and is lower than OODNet. The performance in terms of PSNR of MetaNet is also very close to the performance of OODNet. It can be confidently said, based on the experiments done, that if training time and complexity are considered, similar quality performance can be achieved with MetaNet at a reduced complexity and training cost. Training a single model is sufficient for MetaNet whereas multiple modules need to trained for OODNet.

4.5 Summary

Existing blind JPEG image deblocking networks are complex in terms of execution time and training cost. In this chapter, the use of meta-learning is proposed in order to train a single network that is capable of generating high-quality images from images compressed with QF values not used during the network training process. This single network can handle a higher range of QF values and produce sharp images. The effectiveness of the training algorithm is shown through multiple ablation studies. The comparative studies

Table 4.10: Comparison between OODNet and MetaNet

Comparison Between OODNet and MetaNet					
Dataset	Settings	OODNet		MetaNet	
Live1	QF	PSNR	SSIM	PSNR	SSIM
	5	27.62	0.7444	28.11	0.7573
	15	31.98	0.8717	31.99	0.8713
	25	33.33	0.9013	33.39	0.9030
	35	34.25	0.9184	34.24	0.9186
	QF	PSNR	SSIM	PSNR	SSIM
	10	30.80	0.8389	30.74	0.8374
	20	32.81	0.8902	32.80	0.8903
	30	33.89	0.9121	33.87	0.9120
	40	34.54	0.9234	34.50	0.9231
Classic5	QF	PSNR	SSIM	PSNR	SSIM
	5	28.55	0.7678	28.54	0.7599
	15	32.24	0.8643	32.35	0.8642
	25	33.60	0.8891	33.66	0.8907
	35	34.41	0.9039	34.37	0.9038
	QF	PSNR	SSIM	PSNR	SSIM
	10	31.21	0.8380	31.13	0.8359
	20	33.14	0.8806	33.12	0.8806
	30	34.11	0.8986	34.07	0.8982
	40	34.65	0.9083	34.61	0.9078

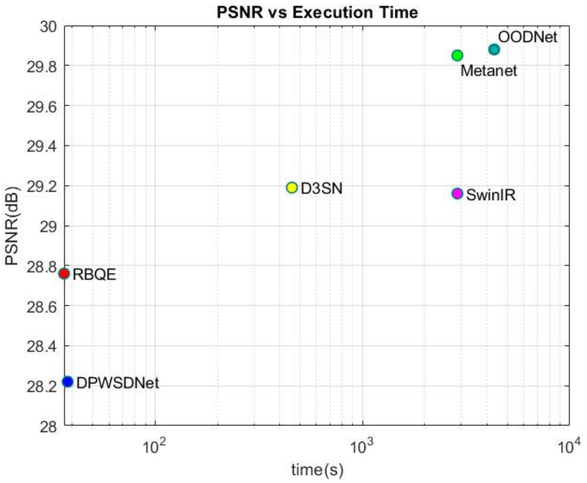


Figure 4.7: Plot of average performance versus the complexity of different image deblocking schemes in the cases of indistribution and out-of-distribution QF values

also show the effectiveness compared to the state-of-the-art networks. A comparative study between OOD and MetaNet is also done. The network OODNet, proposed in Chapter. 3 deblocks produces high-quality images from the compressed images but it is more complex and execution time is higher. MetaNet proposed in this chapter solves the complexity issue and the execution time of this network is same as SwinIR [32], as no modifications are made in the network architecture.

Chapter 5

Conclusion

5.1 Concluding Remarks

Existing deep learning-based pseudo-blind JPEG deblocking schemes perform well on deblocking JPEG images compressed by a QF or a set of QFs used for training the model. The performance drops significantly if the input image falls outside the QF values unseen to the deep deblocking networks during training. This makes deep JPEG deblocking a challenge as it is time and resource-consuming to train models with every QF values separately.

This problem has been addressed in this thesis using in and out-of-distribution and meta-learning. When a QF value falls outside the QF values used to train a model, it is called an out-of-distribution QF value. In Chapter 3, a deep deblocking scheme referred to as OODNet is proposed to deal with images compressed using QF values not used while training the network. The scheme of OODNet consists of four modules: image QF detection module, out-of-distribution detection module, deep deblocking module, and image quality assessment module. As seen the comparative study in Chapter 3, these four modules enabled the network to perform better deblocking than the existing state-of-the-art networks.

In Chapter 4, deep JPEG image deblocking has been addressed from a meta-learning point of view. Though OODNet performs better than state-of-the-art networks, it is computationally expensive, has a higher execution time and requires more memory. MetaNet solves the complexity issue by using only one model of a deep deblocking network which is optimally trained for dealing with QF values outside the QF values used for training the network. The algorithm and effectiveness of MetaNet are demonstrated through multiple experiments done in Chapter 4.

A comparison is done between OODNet and Metanet in Table. 4.10. In Fig.4.7, a plot of average performance versus the complexity of different image deblocking schemes are shown in case of in distribution and out-of-distribution QF values. In the diagram, it is seen that even though OODNet produces better performance it comes at a cost of higher complexity and execution time. MetaNet's execution time is similar to SwinIR, but the performance is very similar to that of OODNet. As multiple models of a deep deblocking network are used in OODNet, the memory usage is also higher compared to MetaNet which only uses a single model of a deep deblocking network.

If execution time, computational complexity, and memory usage are not an issue, the scheme of OODNet will be the obvious choice. However, if similar performance is required at a reduced execution time, computational complexity, and memory usage, the scheme of MetaNet will be the choice.

5.2 Scope for Further Investigation

The following are some examples for further investigating the deep JPEG image deblocking schemes proposed in this thesis.

- The OODNet described in Chapter 3 can be modified for better deblocking performance by changing the deblocking network in the deep deblocking module. A modified version of SwinIR [32] used as the deep deblocking network in the deep deblocking module can be replaced with another network like SwinFIR [60].
- For MetaNet, SwinIR has been used as the deep doblocking network. SwinIR can be replaced using another deep deblocking network to improve the overall performance of the MetaNet scheme.

References

[1] T. Chen, H. Wu, and B. Qiu, "Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 5, pp. 594–602, 2001.

[2] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive dct for high-quality deblocking of compressed color images," in *2006 14th European Signal Processing Conference*, 2006, pp. 1–5.

[3] S. Wu, H. Yan, and Z. Tan, "An efficient wavelet-based deblocking algorithm for highly compressed images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 11, pp. 1193–1198, 2001.

[4] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 576–584.

[5] Y. Tang, X. Zhang, and X. Zhang, "An efficient lightweight network for single image super-resolution," *Journal of Visual Communication and Image Representation*, vol. 93, p. 103834, 2023.

[6] F. Liu, X. Yang, and B. De Baets, "A deep recursive multi-scale feature fusion network for image super-resolution," *Journal of Visual Communication and Image Representation*, vol. 90, p. 103730, 2023.

[7] Q. Zhang, B. Chen, X. Lu, and Q. Xia, “Super-resolution of single multi-color image with guided filter,” *Journal of Visual Communication and Image Representation*, vol. 58, pp. 277–284, 2019.

[8] D. Zhang, J. Shao, Z. Liang, L. Gao, and H. T. Shen, “Large factor image super-resolution with cascaded convolutional neural networks,” *IEEE Transactions on Multimedia*, vol. 23, pp. 2172–2184, 2020.

[9] M. Li, Z. Zhang, J. Yu, and C. W. Chen, “Learning face image super-resolution through facial semantic attribute transformation and self-attentive structure enhancement,” *IEEE Transactions on Multimedia*, vol. 23, pp. 468–483, 2020.

[10] Y. Liu, S. Wang, J. Zhang, S. Wang, S. Ma, and W. Gao, “Iterative network for image super-resolution,” *IEEE Transactions on Multimedia*, vol. 24, pp. 2259–2272, 2021.

[11] A. Esmaeilzehi, M. O. Ahmad, and M. Swamy, “Srnssi: a deep light-weight network for single image super resolution using spatial and spectral information,” *IEEE Transactions on Computational Imaging*, vol. 7, pp. 409–421, 2021.

[12] T. Pang, H. Zheng, Y. Quan, and H. Ji, “Recorrupted-to-recorrupted: Unsupervised deep learning for image denoising,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2043–2052.

[13] S. Anwar and N. Barnes, “Real image denoising with feature attention,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3155–3164.

[14] S. Gu, Y. Li, L. V. Gool, and R. Timofte, “Self-guided network for fast image denoising,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2511–2520.

[15] F. Jia, W. H. Wong, and T. Zeng, "Ddunet: Dense dense u-net with applications in image denoising," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 354–364.

[16] S. Yu, B. Park, and J. Jeong, "Deep iterative down-up cnn for image denoising," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2095–2103.

[17] B. Park, S. Yu, and J. Jeong, "Densely connected hierarchical network for image denoising," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2104–2113.

[18] X. Jia, S. Liu, X. Feng, and L. Zhang, "Focnet: A fractional optimal control network for image denoising," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6054–6063.

[19] J. Ma, C. Peng, X. Tian, and J. Jiang, "Dbdnet: a deep boosting strategy for image denoising," *IEEE Transactions on Multimedia*, vol. 24, pp. 3157–3168, 2021.

[20] R. Ma, S. Li, B. Zhang, and Z. Li, "Towards fast and robust real image denoising with attentive neural network and pid controller," *IEEE Transactions on Multimedia*, vol. 24, pp. 2366–2377, 2021.

[21] Z. Jin, M. Z. Iqbal, D. Bobkov, W. Zou, X. Li, and E. Steinbach, "A flexible deep cnn framework for image restoration," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 1055–1068, 2019.

[22] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017.

[23] A. Esmaeilzehi, M. O. Ahmad, and M. Swamy, “Deep jpeg image deblocking using residual maxout units,” in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 2681–2685.

[24] A. Esmaeilzehi, M. O. Ahmad, and M. Swamy, “Development of new fractal and non-fractal deep residual networks for deblocking of jpeg decompressed images,” in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1271–1275.

[25] J. Guo and H. Chao, “Building dual-domain representations for compression artifacts reduction,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 628–644.

[26] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[27] C. Chen, Z. Xiong, X. Tian, and F. Wu, “Deep boosting for image denoising,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–18.

[28] J. He, X. He, M. Zhang, S. Xiong, and H. Chen, “Deep dual-domain semi-blind network for compressed image quality enhancement,” *Knowledge-Based Systems*, vol. 238, p. 107870, 2022.

[29] Y. Kim, J. W. Soh, J. Park, B. Ahn, H.-S. Lee, Y.-S. Moon, and N. I. Cho, “A pseudo-blind convolutional neural network for the reduction of compression artifacts,” *IEEE Transactions on circuits and systems for video technology*, vol. 30, no. 4, pp. 1121–1135, 2019.

[30] H. Chen, X. He, L. Qing, S. Xiong, and T. Q. Nguyen, “Dpw-sdnet: Dual pixel-wavelet domain deep cnns for soft decoding of jpeg-compressed images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 711–720.

[31] Q. Xing, M. Xu, T. Li, and Z. Guan, “Early exit or not: Resource-efficient blind quality enhancement for compressed images,” in *European Conference on Computer Vision*. Springer, 2020, pp. 275–292.

[32] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, “Swinir: Image restoration using swin transformer,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1833–1844.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[35] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[36] A. Esmaeilzehi, M. O. Ahmad, and M. N. S. Swamy, “Fpnet: A deep light-weight interpretable neural network using forward prediction filtering for efficient single image super resolution,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1937–1941, 2022.

[37] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," *CoRR*, vol. abs/1608.00367, 2016. [Online]. Available: http://arxiv.org/abs/1608.00367

[38] A. Esmaeilzehi, M. O. Ahmad, and M. Swamy, "Ultralight-weight three-prior convolutional neural network for single image super resolution," *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 6, pp. 1724–1738, 2023.

[39] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018. [Online]. Available: http://arxiv.org/abs/1803.08375

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[41] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[42] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[43] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[44] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 7, pp. 2480–2495, 2021.

[45] A. Esmaeilzehi, M. O. Ahmad, and M. Swamy, “Srnmfrb: A deep light-weight super resolution network using multi-receptive field feature generation residual blocks,” in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[46] A. Esmaeilzehi, M. O. Ahmad, and M. N. S. Swamy, “Phmnet: A deep super resolution network using parallel and hierarchical multi-scale residual blocks,” in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[47] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising,” *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[48] S. S. Ahsan, A. Esmaeilzehi, and M. O. Ahmad, “Oodnet: A deep blind jpeg image compression deblocking network using out-of-distribution detection,” *Under review for publication in Journal of Visual Communication and Image Representation*, 2024.

[49] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1122–1131.

[50] H. Sheikh, “Live image quality assessment database release 2,” *http://live. ece. utexas. edu/research/quality*, 2005.

[51] A. Foi, V. Katkovnik, and K. Egiazarian, “Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images,” *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1395–1411, 2007.

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

[53] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[54] A. Mittal, A. K. Moorthy, and A. C. Bovik, “No-reference image quality assessment in the spatial domain,” *IEEE Transactions on image processing*, vol. 21, no. 12, pp. 4695–4708, 2012.

[55] A. Mittal, R. Soundararajan, and A. C. Bovik, “Making a “completely blind” image quality analyzer,” *IEEE Signal processing letters*, vol. 20, no. 3, pp. 209–212, 2012.

[56] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte, “Plug-and-play image restoration with deep denoiser prior,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6360–6376, 2022.

[57] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, “Residual non-local attention networks for image restoration,” *CoRR*, vol. abs/1903.10082, 2019. [Online]. Available: http://arxiv.org/abs/1903.10082

[58] S. S. Ahsan, A. Esmaeilzehi, and M. O. Ahmad, “Metanet: A deep blind jpeg image compression deblocking network using meta learning,” *To be submitted for journal publication*.

[59] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” *CoRR*, vol. abs/1703.03400, 2017. [Online]. Available: http://arxiv.org/abs/1703.03400

[60] D. Zhang, F. Huang, S. Liu, X. Wang, and Z. Jin, “Swinfir: Revisiting the swinir with fast fourier convolution and improved training for image super-resolution,” *arXiv preprint arXiv:2208.11247*, 2022.