Edge-Aware Image Compression using Deep Learning-based Super-resolution Network

Dipti Mishra, *Graduate Student Member, IEEE,* Satish Kumar Singh, *Senior Member, IEEE,* Rajat Kumar Singh, *Senior Member, IEEE,* and Krishna Preetham

Abstract—We propose a learning-based compression scheme that envelopes a standard codec between pre and post-processing deep CNNs. Specifically, we demonstrate improvements over prior approaches utilizing a compression-decompression network by introducing: (a) an edge-aware loss function to prevent blurring that is commonly occurred in prior works & (b) a super-resolution convolutional neural network (CNN) for post-processing along with a corresponding pre-processing network for improved rate-distortion performance in the low rate regime. The algorithm is assessed on a variety of datasets varying from low to high resolution namely Set 5, Set 7, Classic 5, Set 14, Live 1, Kodak, General 100, CLIC 2019. When compared to JPEG, JPEG2000, BPG, and recent CNN approach, the proposed algorithm contributes significant improvement in PSNR with an approximate gain of 20.75%, 8.47%, 3.22%, 3.23% and 24.59%, 14.46%, 10.14%, 8.57% at low and high bit-rates respectively. Similarly, this improvement in MS-SSIM is approximately 71.43%, 50%, 36.36%, 23.08%, 64.70% and 64.47%, 61.29%, 47.06%, 51.52%, 16.28% at low and high bit-rates respectively. With CLIC 2019 dataset, PSNR is found to be superior with approximately 16.67%, 10.53%, 6.78%, and 24.62%, 17.39%, 14.08% at low and high bit-rates respectively, over JPEG2000, BPG, and recent CNN approach. Similarly, the MS-SSIM is found to be superior with approximately 72%, 45.45%, 39.13%, 18.52%, and 71.43%, 50%, 41.18%, 17.07% at low and high bit-rates respectively, compared to the same approaches. A similar type of improvement is achieved with other datasets also.

Index Terms—Edge aware loss, deep learning, CNN, compression-decompression, edge detector, HED, super-resolution network, codec compatible

I. INTRODUCTION

EVERY day an enormous amount of data is continuously transmitted and stored. Mostly the transmission happens through the web, and the images on the internet are of high resolution and larger in size. So to improve the efficacy of the transmission and storage, reducing the size of these files is a compulsion and an issue of big concern. Hence, image compression is widely used for storage & transmission which aims at reducing the different types of redundancies present in the image. A higher compression ratio reduces the size of the image to be stored as bits-per-pixel (bpp) allotted decreases, but it heavily affects the reconstruction quality of the image due to the loss that occurred during compression. So, there is a trade-off between the compression ratio and the quality of the image. Among alternative compression formats, the Joint Photographic Experts Group (JPEG) [1] standard is by-far

D. Mishra, R. K. Singh, are with the Department of Electronics & Communication Engineering, Indian Institute of Information Technology, Allahabad, Prayagraj, India (e-mail: {rse2017502, rajatsingh}@iiita.ac.in)

S. K. Singh, K. Preetham are with the Department of Information Technology, Indian Institute of Information Technology, Allahabad, Prayagraj (sk.singh@iiita.ac.in, krishna.preetham09@gmail.com)

dominant; currently, almost all pictorial images are stored and communicated in JPEG format. Given the extensive utilization of JPEG, modern SmartPhone & tablet devices commonly include hardware support for JPEG compression & decompression. Methods that can improve compression performance, while maintaining compatibility with the JPEG format, are, therefore, of strong research interest. Theoretically, JPEG, the standard algorithm, which is a discrete cosine transform (DCT) based emphasizes low-frequency components while processing the image for the task of image compression. JPEG2000 [2], on the other hand, being lossless & lossy both, is a region of interest (ROI) based compression scheme based on multi-scale decomposition into sub-bands through discrete wavelet transform (DWT). As compared to JPEG, JPEG2000 produces only ringing effects near the edges in the image. However, at high bit-rates, these artifacts become less visible or almost imperceptible. Later on, many techniques like BPG[1] & WebP[2] came into existence which provided a significant quantitative & qualitative improvement with respect to reconstructed image quality. Nowadays, deep learning is producing interesting & eye-catching results in the field of biometrics & computer vision, it has started proving itself in the image compression domain also. Some of the best compression algorithms are discussed in the next sections.

A. Deblocking and Post-processing methods

Various post-processing modules have been proposed in the literature. Modeling of the image-prior, based on maximum a posteriori (MAP) criteria is used in [3], but it is found to be quite more expensive than handcrafted models. The post-processing method in [4] based on the adaptive DCT method is quite good, but failed to generate sharp images. The notion of Wiener filtering for denoising used in [5] was novel, however, the images generated contained highly visible noise at the edge regions. Similarly, the modeling of image prior and quantization is found to be expensive in [6]. Ren et al. [7] combined the local sparsity image prior model and non-local similarity model for artifacts reduction using low-rank minimization and patch clustering proposed in [3] and [6] respectively. Dictionary learning and variational regularization used in [8] for restoring images outperformed the other decompression method but consumed substantial computational time. Weighted nuclear norm minimization (WNNM) concept used in [9] being complex, outperformed Dabov et al. [5] approach while preserving the local structure and reducing visual artifacts taking more computation time.

[1]https://bellard.org/bpg/
[2]https://developers.google.com/speed/webp/

The deblocking method, given in [10] using a non-convex low rank constrained model was a good optimization of the problem. The denoising method proposed by Zhang et al. [11] helped in reducing artifacts but with the lack of retaining edge information in reconstructed images. The benchmark deblocking method ARCNN, proposed by Dong et al. [12] outperformed the above-discussed methods used for image compression. All these deblocking and post-processing models aimed to improve the reconstruction quality while ignoring the joint optimization of the traditional codec with encoder-decoder structure.

B. Down and Upsampling based CNN methods

Recent work [13], [14] has demonstrated promising results with deep learning methodologies for compression. Specifically, the work in [13] proposed an "end-to-end" compression framework wherein pre & post-processing DNN is introduced around a standard compression codec. However, the training procedure did not take care of high-frequency components (i.e. edges) due to optimization with mean square error (MSE) resulting in smooth reconstructed images. Moreover, the author did not present the rate-distortion analysis for the JPEG codec, which is needed to compare the compression performance based on bit-rates. Like [13], the approach in [14] uses jointly trained CNNs for the pre-processing and post-processing stages. These stages incorporate resolution reduction only for low JPEG quality factor (QF) and maintain resolution for higher QF. On the other hand, the work in [15] is based on investigating different downsampling networks and super-resolving or upsampling networks along with the different training strategies. The sequential training of downsampling CNN with upsampling CNN only has been considered, without the consideration of any codec in-between.

C. Recent State-of-the-Art Algorithms

In 2016, Toderici et al. [16] introduced an RNN based full resolution compression exploiting the principle of progressive encoding-decoding of images. In 2017, Balle et al. [17] proposed to use generalized divisive normalization based local gain control method with the use of additive uniform noise to make non-differentiable quantization function a continuous function. Theis et al. [18] developed an algorithm to use a single autoencoder for providing variable bit-rate image compression by just tweaking with the size of latent space representation before quantization. Basically, the work in [16], [19], [20] is based on the calculation of the probability of binary representation for context-adaptive coding. Basically, Johnston et al. [20] extracted binary information for maximizing the entropy. In 2018, Li et al. [21] came up with a compression model based on content prediction with the help of importance map generation through CNN. Later on, Balle et al. extended the work in [17] by exploiting spatial dependencies for variable rate image compression in [22]. Similarly, Lee et al. [23] extended the work in [22] by proposing a context-adaptive entropy model by incorporating bit-consuming and bit-free contexts. On the other hand, the approach reported by Minnen et al. [24] was based on

weighted auto-regressive and hierarchical priors for contextual image compression. Mentzer et al. [25] proposed to focus on rate-distortion trade-offs by learning a conditional probability model of latent space distribution in a 3D-CNN autoencoder. Later on in 2019's, Choi et al. [26] proposed a variable bit-rate image compression in the hierarchy of having one compression module for providing multiple bit-rates. In early 2020's, the model proposed by Yang et al. [27] was based on multi-layer feature modulation which reduced the problem of lower performance at low bit-rates. Chen et al. [28] proposed to learn a single CNN model at high bit-rates and then scaling the bottleneck by some factor for low bit-rates.

Motivated by the discussions on downsampling/upsampling based CNN methods, we utilize an "end-to-end" deep compression framework compatible with the traditional codec by utilizing pre & post-processing DNNs. Different from prior works, however, we:

- introduce an edge-aware loss function during compression, which significantly improves compression performance. The edge-aware loss function is motivated by the need to minimize edge distortion in compression, which the MSE loss function in [13] does not emphasize.
- for the low bit-rate regime, utilize a super-resolution network in the post-processing stage.
- prefer to use direct image-to-image learning over patch-to-patch learning criteria which is generally practiced in previous compression algorithms. The later scheme degrades the training performance, prediction efficiency, and lead to distortions in the output image.

In the proposed approach, we utilize the enhanced deep super-resolution (EDSR) network [29], which is a multi-scale network designed for specific super-resolution scaling that reduces model size and training time using residual learning compared with alternative approaches. The residual learning used in the super-resolution network helps in preserving the detail components. Because the quantization in the codec represents a non-differentiable operation, we use progressive training that co-optimizes the pre & post-processing modules in an "end-to-end" manner by appropriately approximating the codec. The super-resolution is chosen to improve the quality of the JPEG-compressed output by reducing blocking artifacts at the time of post-processing. In-fact, super-resolution which is a process of converting a low-resolution image into its high-resolution counterparts is exactly equivalent to image decompression which aims to improve the quality of the decoded image by adding more high-frequency details. Hence, the image super-resolution mechanism can be called a special case of an image compression-decompression algorithm. Different from [13], [14], the proposed network uses an edge-aware loss function to optimize the pre-processing module. In addition to that, it uses EDSR module instead of VDSR [30] module (used in [13]) & 5 layer encoder-decoder network (used in [14]) as post-processing module. The proposed algorithm is quite different from the work reported in [14], [15] in terms of the pre-processing module and post-processing module exploited. Moreover, the algorithm presented in [14], [15] exploited switching between one network to another when
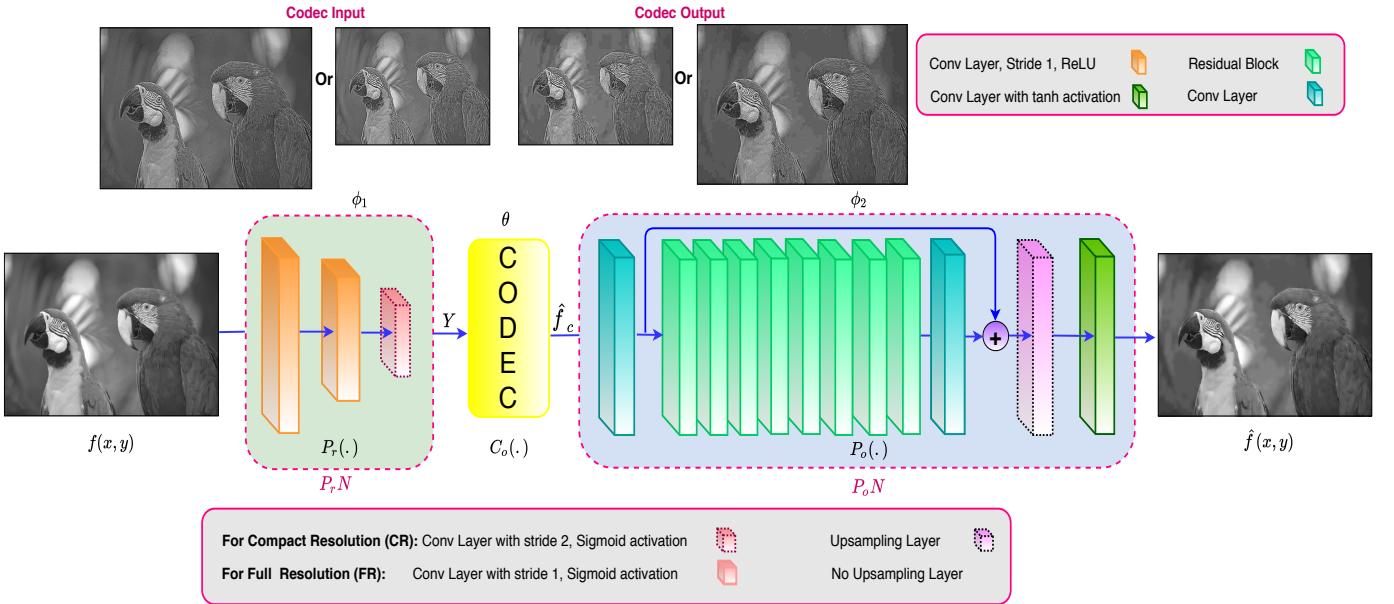
Fig. 1. Detailed architecture of the proposed compression-decompression network.

investigating compression performance from low bit-rates to high bit-rates, which does not seem optimal. Also, the author did not compare the work reported in [15] with any DNN based image compression algorithms.

The remaining sections of the paper are organized as follows. Section II presents the proposed approach covering details of the post-processing DNN & the edge-aware loss function [31], [32] that we introduce. Section III discusses for the datasets used and other experimental setup details. In Section IV, comparison between different experiment strategies and modules has been shown. Section V throws light on the comparison of the final proposed algorithm with the state-of-the-art algorithms. Section VI finally summarizes the effectiveness of the proposed algorithm reported in this paper.

II. EDGE-AWARE CODEC COMPRESSION

The proposed architecture with complete layer-wise details for the three separate modules is shown in Fig. 1. The image to be compressed, $\mathbf{f}(x, y)$, goes through an image pre-processing network (P_rN) that generates the input image $\mathbf{Y} = P_r(\mathbf{f}, \boldsymbol{\phi_1})$ for the standard codec (here, JPEG), where $\boldsymbol{\phi_1}$ are the network weights, i.e, the learned parameters, of P_rN. Decompression is performed by first decoding the image from its coded JPEG representation to obtain the image $\hat{\mathbf{f}}_c = C_o(\mathbf{Y}, \theta)$, where, $Co(.)$ represents the codec function, θ denotes the JPEG QF. The reconstructed image $\hat{f}(x, y)$ is then obtained by post-processing the image $\hat{\mathbf{f}}_c$ through the post-processing network (P_oN); this operation is represented as $\hat{\mathbf{f}}(x, y) = P_o(\hat{\mathbf{f}}_c, \boldsymbol{\phi_2})$, where $\boldsymbol{\phi_2}$ are the network weights of P_oN. In this process: (a) for P_E (CR) (proposed edge-aware compression algorithm based on compact resolution (CR) network), the image $\mathbf{Y}$ produced by P_rN is a CR (scaled down in size by 2 along each dimension by using a stride of 2 in the final convolutional layer), & P_oN includes an upsampling layer that scales up the resolution by a factor of 2 along each dimension & (b)

for P_E (FR) (proposed edge-aware compression algorithm based on full resolution (FR) network), the resolution is held constant by the processing by P_rN & P_oN. P_rN uses an arrangement of layers identical to [13], except that the second convolutional layer consists of 32 filters instead of 64 filters in [13]. Accordingly, to make the downsampled image to be as informative as possible, P_rN uses stride 2 in the last CNN layer (different from Li's). We defer using the batch normalization (BN) (as used in [13]) for better generalizability, stable training, and consistent performance. Firstly, normalizing features with the mean and variance of batch during training generate artifacts, if the statistical characteristics of the training and testing datasets largely differ [29], [33]. Secondly, in PSNR oriented tasks, like image enhancement, image super-resolution & image compression, BN gives rise to unpleasing artifacts when the CNN network is deeper. Thirdly, feature normalization reduces range flexibility which is also experimentally proven in [29]. Lastly, BN layers consume the same amount of memory as the preceding convolutional layers, this elimination also significantly reduces the network parameters & GPU memory usage. The post-processing network P_oN uses an adaptation of the EDSR network [29], with an up-sampling layer at the output when a CR representation is used. The use of this upsampling layer is dropped when the FR representation is used. The upsampling layer in CR representation replaces the bicubic interpolation that was used in [13] to recover a full-size image from CR representation. Fully connected layers have not been used in the proposed network making it a fully convolutional network (FCN) in nature.

We stress upon the fact that, while the approach of reducing the codec input to a lower (compact) resolution is beneficial at low rates (correspondingly, lower reconstruction quality), at higher rates, better rate-distortion performance is obtained by retaining the original or full resolution instead of using the
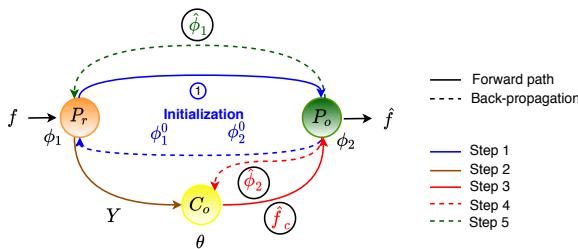
Fig. 2. Learning flow for proposed algorithm.

reduced resolution.

Structural and textural details are the important features that are highly correlated to human visual perception. Training a DNN with MSE produces smooth images at the output while ignoring the fine structural features, which are also important details for human visual perception. The edges present in the images which are mainly responsible for structural features should also be considered and given some weight while training the network. Accordingly, the edge-aware loss function is defined as,

$$\mathscr{L}_{edge-aware} = \alpha \times \mathscr{L}_{MSE} + \gamma \times \mathscr{L}_{edge\ weighted\ MSE} \quad (1)$$

where, $\gamma = 1 - \alpha$ & α decides the weight for the two components in Eq. (1). The first component in Eq. (1) is basically MSE and the second component is the edge-preserving loss function (weighted MSE) which helps to restore edges or structural information in the image. Learning high-quality edges ensures the good quality of the reconstructed images.

Accordingly, for the P_rN module, a novel edge aware loss function for an image $\mathbf{f}$ is introduced as in Eq. (2),

$$\mathscr{L}_r(\boldsymbol{\phi_1}, \mathbf{f}) = \alpha \left\| P_o(\hat{\boldsymbol{\phi_2}}, P_r(\boldsymbol{\phi_1}, \mathbf{f})) - \mathbf{f} \right\|_2^2$$
$$+ \gamma \left\| \mathbf{E} \circ \left(P_o(\hat{\boldsymbol{\phi_2}}, P_r(\boldsymbol{\phi_1}, \mathbf{f})) - \mathbf{f} \right) \right\|_2^2 \quad (2)$$

where $\|\cdot\|_2$ denotes the ℓ-2 vector norm, $\mathbf{E}$ denotes the edge weight map (binary map representing edge at 1 & no edge at 0) obtained by applying the edge detector to the input image, & $\circ$ denotes pixel-wise multiplication.

The post-processing network P_oN uses the MSE loss function as given in Eq. (3) as,

$$\mathscr{L}_o(\boldsymbol{\phi_2}, \mathbf{f}) = \left(\left\| P_o(\boldsymbol{\phi_2}, C_o(P_r(\hat{\boldsymbol{\phi_1}}, \mathbf{f}), \theta)) - \mathbf{f} \right\|_2^2 \right) \quad (3)$$

Also, Fig. 2 clearly shows the progressive learning workflow for the proposed algorithm. Initially, both the CNN modules are initialized with the learning parameters $\boldsymbol{\phi_1^0}$ & $\boldsymbol{\phi_2^0}$ with training as a simple autoencoder, as shown in step 1. Then, the CR (in case of P_E (CR)) or FR representation (in case of P_E (FR)) is predicted using weights of P_rN module as in step 2. This representation is compressed with JPEG for a particular QF, as in step 3. Now, we define any arbitrary variable $\hat{f}_c$, codec output, acting as input to P_oN module. Accordingly, for an image $\mathbf{f}$, codec output $\hat{\mathbf{f}}_c$ can be written as in Eq. (4) as,

$$\hat{\mathbf{f}}_c = C_o(P_r(\hat{\boldsymbol{\phi_1}}, \mathbf{f}), \theta) \quad (4)$$

Hence, further to optimize P_oN module for batch of images, the learning parameter $\boldsymbol{\phi_2}$ can be obtained as in Eq. (5),

$$\hat{\boldsymbol{\phi_2}} = \arg\min_{\boldsymbol{\phi_2}} \left(\sum_{k=1}^{N} \left\| P_o(\boldsymbol{\phi_2}, C_o(P_r(\hat{\boldsymbol{\phi_1}}, \mathbf{f}_k), \theta)) - \mathbf{f}_k \right\|_2^2 \right) \quad (5)$$

For batch of images, Eq. (5) can be modified using using Eq. (4) to,

$$\hat{\boldsymbol{\phi_2}} = \arg\min_{\boldsymbol{\phi_2}} \left(\sum_{k=1}^{N} \left\| P_o(\boldsymbol{\phi_2}, \hat{\mathbf{f}}_{c_k}) - \mathbf{f}_k \right\|_2^2 \right) \quad (6)$$

where Eq. (6), implies that the codec output $\hat{\mathbf{f}}_{c_k}$ in Eq. (4) and ground truth $\mathbf{f}_k$ are then used to optimize the P_oN training parameter $\boldsymbol{\phi_2}$ as in step 4. With $\hat{\boldsymbol{\phi_2}}$, P_rN module is optimized by back-propagating error through P_rN module to obtain $\hat{\boldsymbol{\phi_1}}$ in step 5 as in Eq. (7) as,

$$\hat{\boldsymbol{\phi_1}} = \arg\min_{\boldsymbol{\phi_1}} \left(\alpha \sum_{k=1}^{N} \|A\|_2^2 \right) + \arg\min_{\boldsymbol{\phi_1}} \left(\gamma \sum_{k=1}^{N} \|\mathbf{E} \circ A\|_2^2 \right) \quad (7)$$

where, A in Eq. (7) can be written as,

$$A = P_o(\hat{\boldsymbol{\phi_2}}, C_o(P_r(\boldsymbol{\phi_1}, \mathbf{f}_k), \theta)) - \mathbf{f}_k \quad (8)$$

Since to optimize the weights of P_rN, codec is excluded in the back-propagation path, accordingly, on putting the value of A, finally it can be deduced that,

$$\hat{\boldsymbol{\phi_1}} = \arg\min_{\boldsymbol{\phi_1}} \left(\alpha \sum_{k=1}^{N} \left\| P_o(\hat{\boldsymbol{\phi_2}}, P_r(\boldsymbol{\phi_1}, \mathbf{f}_k)) - \mathbf{f}_k \right\|_2^2 \right)$$
$$+ \arg\min_{\boldsymbol{\phi_1}} \left(\gamma \sum_{k=1}^{N} \left\| \mathbf{E} \circ \left(P_o(\hat{\boldsymbol{\phi_2}}, P_r(\boldsymbol{\phi_1}, \mathbf{f}_k)) - \mathbf{f}_k \right) \right\|_2^2 \right) \quad (9)$$

Note that codec is used in the forward path only to predict the input to P_oN. The flow can also be called predicting (testing) while training. In every epoch, the weights of P_oN are fine-tuned according to the JPEG decoded output at a particular QF. Then the weights of P_rN are fine-tuned by fixing the weights of P_oN. Hence, it is clear that the training of both the CNN modules depends upon JPEG QF θ, as all the intermediate outputs $\mathbf{Y}$ & $\hat{\mathbf{f}}_c$ depends upon θ. Hence, it is deduced that the codec JPEG also plays a role in CNN weight learning of P_rN & P_oN. Here, we have used the JPEG codec, however, the framework is generalized to use any codec.

III. IMPLEMENTATION DETAILS

A. Dataset

The proposed compression-decompression model is trained on 400 images (256×256) of the ImageNet dataset [34]. In order to compare the results with [14], we have assessed the algorithm on various datasets namely Live 1 [35], Set 14 [36], Set 5 [37] & Set 7 (same as built up in [14]). These datasets are having different attributes which also help in the better generalization of the proposed scheme. Live 1 dataset has twenty-nine diversified generic PNG images with sizes ranging from 438×634 to 720×480. These images contain high resolution and high-quality, sharp images.

Set 14 contains fourteen low-frequency PNG images with size ranging from 250×361 to 768×512. Set 5 contains five edge centered luminance images. These are images with high-frequency domination where size varies from 280×280 to 512×512. Apart from these three standard test sets, a test of 7 images named Set 7 is prepared (as in [14]) to compare the results with Zhao's approach [14]. This special set contains both smooth and sharp images with size 256×256, 512×512 & 512×768, and reported as challenging for image compression application. To compare with the recent techniques, the proposed algorithm is also assessed on Kodak benchmark dataset [38] and CLIC challenging dataset [39]. Kodak dataset contains uncompressed true-color PNG images, with a size of 768×512 or 512×768. CLIC 2019 consists of high-resolution and high-quality images with size ranging from 512×768 to 2048×1646. Moreover, to compare the deblockiness effect of the proposed algorithm, we have also tested Classic 5 [40] & General 100 [41] datasets. Classic 5 contains five BMP gray images of 512×512 size. General 100 contains a hundred images in BMP format with zero compression, which makes it highly suitable for the compression task. It contains good quality images with clear sharp edges but fewer smooth regions. The image dimensions range from 131×112 to 710×704.

B. Hyperparameter Tuning

Both the CNN modules are trained for five iterations in each epoch. Adam optimizer is used to optimize the weights of the network that has shown very good performance as reported in [42]. Initially, $\alpha = 0.01$ was selected as the learning rate, but the loss observed was very high. Then we decreased the learning rate to $\alpha = 0.001$, due to which loss also decreased substantially. On further decreasing the learning rate, we did not get much improvement. Hence, we finalized the learning rate at $\alpha = 0.001$. The momentum parameters are selected as $\beta_1 = 0.9$, $\beta_2 = 0.999$ & $\epsilon = 1 \times 10^{-7}$, which are reported to yield good results as in [13], [29], [43] to find the global minima. The batch size depends upon the computational platform used for the implementation of the CNN based algorithm. It should fit into the memory specification of CPU or GPU based architecture. Generally, training on small batch sizes speeds up the learning process at the cost of low accuracy on account of noise in the training process. On the other hand, with large batch size, the learning process converges slowly with better prediction or estimation. Accordingly, we have set a mini-batch size of 10, suitable for the used platform. We have finalized training epochs to be 50, as, after that, the performance was not improving. All the experiments have been conducted using Python with Tensorflow & Keras deep learning framework on TITAN X (Pascal)/PCIe/SSE2 16 GB CPU RAM, 12 GB GPU enabled system.

C. Evaluation Parameters

The original image $\mathbf{f}(x, y)$ has been compared with $\hat{\mathbf{f}}(x, y)$ for the assessment of the proposed compression-decompression algorithm. We have used both objective and

---

[3]The results of Cavigelli et al. [45] are not given at QF=30.
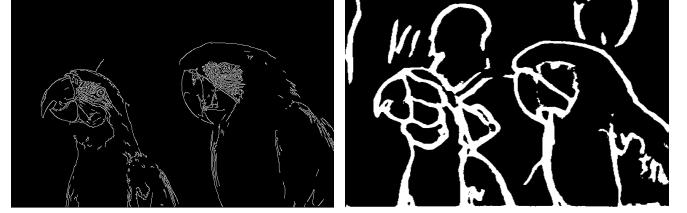

Fig. 3. Comparison of the performance of classical CED and learning-based HED on "Kodim 23" image of Kodak dataset. Clearly, HED produces strong edges and eliminate false edges which are not contextually required.

subjective performance metrics i.e., PSNR and SSIM for a comprehensive assessment. Along with the consideration of structural information, a metric should be chosen which incorporates image details at various resolutions. Therefore, MS-SSIM, a measure based on structural information at different scales is chosen for assessment. In the same direction, we have also calculated PSNRB [47] which incorporates the effect of blocking artifacts (by incorporating the blocking effect factor) while calculating the PSNR. To check for the edge-preserving property of edge-aware loss function we have calculated intersection over union (IoU) to check the overlap regions between true edges and retained edges. Bjontegaard metric calculation [48] has also been conducted to compare the coding gain achieved with the proposed algorithm with the other methods. The BD metric facilitate the comparison of area under rate-distortion curves in terms of the average PSNR improvement or the average percent bit-rate saving. A negative (positive) value of BD-Rate indicates a decrease (increase) of bit-rate for the same PSNR. A positive (negative) value of BD-PSNR indicates an increase (decrease) of PSNR for the same bit-rate.

IV. ABLATION STUDIES

The proposed network has been trained separately for CR & FR representation for a wide range of QFs, i.e., 2, 5, 6, 10, 20, 30, 40, 50, 60, 80, 90 & 100. In the sections below, we justify the formulation of the proposed algorithm based on various aspects/choices available.

A. Comparison of training the pre-processing network separately with Canny edge detector (CED) & learning-based detector (HED)

Many edge detection algorithms are available in the literature. CED is one of the good classical edge detection algorithms which uses the idea of change in intensity on each pixel in an image, i.e., the intensity is very high on the edges. CED identifies edges in 4 steps: noise removal, gradient calculation, non-maximal suppression, and hysteresis thresholding. Since the CED only focuses on local changes and it has no semantic understanding (understanding the content of the image), it has limited accuracy. Semantic understanding is crucial for edge detection that is why learning-based detectors generate better results than CED. Hence, we have separately tested the classical Canny Edge Detection [49] and learning-based edge detector. For learning-based edge detection, we have chosen the Holistically-Nested Edge Detection technique (HED), which has been set as a benchmark technique for

Fig. 4. (a), (c) & (e): P_E (CR); (b), (d) & (f): P_E (FR) compression performance on "Kodim 04" image of Kodak dataset at QF of 6 (row 1), 10 (row 2) & 50 (row 3). Compression performance is (a) 6, 0.042, 27.19, 0.7644, 0.8789, 27.19, 0.1832; (b) 6, 0.1238, 28.79, 0.8466, 0.9116, 28.79, 0.3756; (c) 10, 0.0528, 28.46, 0.7931, 0.9135, 28.46, 0.2071; (d) 10, 0.1832, 32.01, 0.8954, 0.9525, 32.01, 0.4856; (e) 50, 0.1144, 31.71, 0.8554, 0.9656, 31.72, 0.3848; (f) 50, 0.4925, 37.58, 0.9600, 0.9912, 37.58, 0.6839. The four images in each sub figure represents f, Y, $\hat{f}_c$ and $\hat{f}$ respectively. It is clear that FR network performed quite better at high bit-rate & CR network performed better at low bit-rate. Visually, at low rates, the proposed algorithm is also be seen to reduce blocking artifacts. The numbers for each sub figure specify the QF, Rate (bpp), PSNR (dB), SSIM, MS-SSIM, PSNRB (dB) and mIoU.

Fig. 5. *Left:* At QF:10, JPEG decoded image with bpp=0.1026, PSNR=27.78, SSIM=0.7623, MS-SSIM=0.9182, *Middle:* Post-processed image using proposed CR image representation based algorithm with bpp=0.06, PSNR=25.17, SSIM=0.7546, MS-SSIM=0.8896, *Right:* Post-processed image using proposed FR image representation based algorithm with bpp=0.2074, PSNR=29.97, SSIM=0.8892, MS-SSIM=0.9485. Both the CR & FR image representation network produce images with a quite reduction in blocking artifacts by preserving edges & textures as compared to JPEG.

TABLE I
PERFORMANCE EVALUATION (PSNR (DB)/SSIM/MS-SSIM) SHOWING THE EFFICACY OF HED DETECTOR OVER CANNY EDGE DETECTOR, WHEN TESTED ON KODAK DATASET.

QF / Algorithms	10	20	10	20
	Canny Edge Detector (CED)		HED	
$P_E(CR)$	24.57/0.7523/0.8946	24.80/0.7642/0.9055	26.13/0.7679/0.9131	27.63/0.8067/0.9483
$P_E(FR)$	28.99/0.8764/0.9328	30.86/0.9077/0.9633	30.07/0.8983/0.9574	32.24/0.9261/0.9764

TABLE II
PERFORMANCE EVALUATION (PSNR (DB)/SSIM/MS-SSIM) SHOWING THE EFFICACY OF EDSR OVER VDSR POST-PROCESSING MODULE, WHEN TESTED ON KODAK DATASET.

QF / Networks	10	20	30	40
$P_{E_{VDSR}}(CR)$	24.61/0.6262/0.8559	25.59/0.6800/0.9098	26.04/0.7106/0.9304	26.35/0.7296/0.9418
$P_{E_{VDSR}}(FR)$	27.76/0.7700/0.9290	29.79/0.8472/0.9679	31.54/0.8795/0.9791	32.31/0.8983/0.9844
$P_{E_{EDSR}}(CR)$	26.13/0.7679/0.9131	27.63/0.8067/0.9483	28.25/0.8304/0.9612	28.52/0.8365/0.9658
$P_{E_{EDSR}}(FR)$	30.07/0.8983/0.9574	32.24/0.9261/0.9764	33.55/0.9451/0.9862	34.76/0.9507/0.9898

TABLE III
PERFORMANCE EVALUATION (PSNR (DB)/SSIM/MS-SSIM) TO CHECK THE EFFECT OF PRE (P_rN) & POST PROCESSING NETWORKS (P_oN), WHEN TESTED ON KODAK DATASET.

QF / Networks	10	20	30	40
$P_rN + Codec$ (CR)	26.88/0.7419/0.9258	27.68/0.7996/0.9644	27.83/0.8198/0.9758	27.90/0.8288/0.9808
$P_rN + Codec + P_oN$ (CR)	26.13/0.7679/0.9131	27.63/0.8067/0.9483	28.25/0.8304/0.9612	28.52/0.8365/0.9658
$P_rN + Codec$ (FR)	28.01/0.7702/0.9291	30.18/0.8466/0.9678	31.60/0.8801/0.9795	32.53/0.8990/0.9849
$Codec + P_oN$ (CR/FR)	28.96/0.7952/0.9413	31.22/0.8630/0.9717	32.30/0.8900/0.9810	33.17/0.9061/0.9856
$P_rN + Codec + P_oN$ (FR)	30.07/0.8983/0.9574	32.24/0.9261/0.9764	33.55/0.9451/0.9862	34.76/0.9507/0.9898

TABLE IV
MEAN INTERSECTION OVER UNION SHOWING EFFECTIVENESS OF EDGE AWARE LOSS FUNCTION OVER MSE LOSS FUNCTION AT QF:10 & 50. P_M: PROPOSED ALGORITHM WITH TRAINING OF BOTH MODULES ON MSE LOSS, P_E: PROPOSED ALGORITHM WITH TRAINING OF P_rN & P_oN MODULES ON EDGE AND MSE LOSS RESPECTIVELY.

Dataset / Algorithm	Set7	Set14	Live 1	Set5	Kodak	CLIC 2019	General 100	Classic 5	Average
QF					10				
P_M (CR)	0.3571	0.2847	0.2503	0.3063	0.2343	0.2496	0.2952	0.2734	0.2814
P_M (FR)	0.5775	0.5163	0.5063	0.5105	0.4970	0.4816	0.5373	0.5066	0.5225
P_E (CR)	0.3589	0.2815	0.2539	0.3000	0.2375	0.2531	0.2993	0.2739	0.2823
P_E (FR)	0.6068	0.5279	0.5322	0.5347	0.5252	0.5020	0.5575	0.5264	0.5391
QF					50				
P_M (CR)	0.5332	0.4418	0.4006	0.4880	0.3868	0.4185	0.4853	0.4246	0.4474
P_M (FR)	0.6413	0.5776	0.5622	0.5393	0.5478	0.5362	0.6093	0.5754	0.5736
P_E (CR)	0.5154	0.4257	0.3900	0.4613	0.3771	0.4041	0.4612	0.4282	0.4329
P_E (FR)	0.7676	0.7066	0.7137	0.7134	0.7031	0.6984	0.7393	0.7257	0.7209

TABLE V
PERFORMANCE EVALUATION (PSNR (DB)/SSIM/MS-SSIM) SHOWING THE EFFECTIVENESS OF EDGE AWARE LOSS FUNCTION OVER MSE LOSS FUNCTION, WHEN TESTED ON KODAK DATASET.

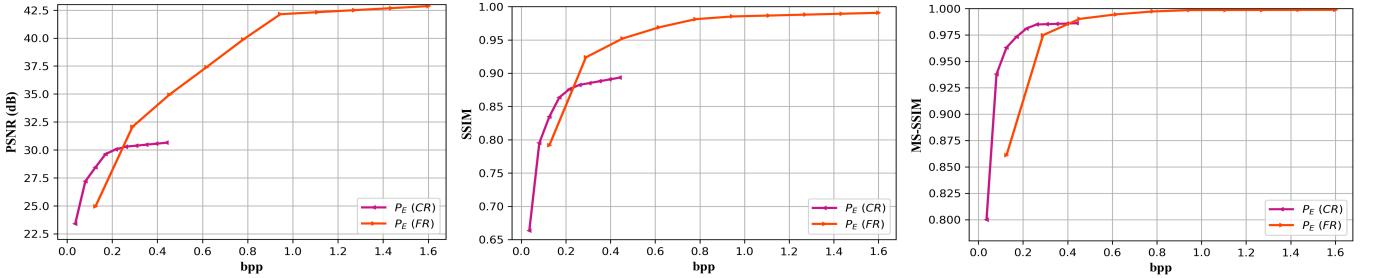| QF / Algorithms	10	20	30	40
$P_M(CR)$	24.30/0.7596/0.8959	26.55/0.8048/0.9463	27.04/0.8131/0.9542	26.65/0.8181/0.9644
$P_E(CR)$	26.13/0.7679/0.9131	27.63/0.8067/0.9483	28.25/0.8304/0.9612	28.52/0.8365/0.9658
$P_M(FR)$	26.09/0.8887/0.9464	31.77/0.8343/0.9585	31.86/0.9230/0.9660	33.67/0.9415/0.9709
$P_E(FR)$	30.07/0.8983/0.9574	32.24/0.9261/0.9764	33.55/0.9451/0.9862	34.76/0.9507/0.9898

Fig. 6. Performance comparison showing the applicability of proposed CR & FR networks based algorithm.

TABLE VI
PERFORMANCE COMPARISON (PSNR(DB)/SSIM/MS-SSIM) SHOWING EFFECT OF CNN LAYER WITH STRIDE 2 IN $P_r N$, WHEN TESTED ON KODAK DATASET.

QF Stride 2	10	20
At first layer	26.33/0.7372/0.9095	27.55/0.7876/0.9436
At last layer	26.13/0.7679/0.9131	27.63/0.8067/0.9483

edge detection after the development in the CNN architectures. HED is a learning-based end-to-end edge detection system that uses a trimmed VGG-like CNN for an image to image prediction task. It is based on multi-scale and multi-level feature learning and claims to learn rich hierarchical features which are crucial. However, Canny edges are not directly connected and produce some spatial shifts and inconsistencies. It makes use of the side outputs of intermediate layers to learn edges synchronizing with the contextual information leaving undesired and false edges. Since the feature maps generated at each layer are of different sizes, it effectively looks at images at different scales. Generally, edge detectors are supposed to assume 90% of the ground truth as non-edge, but HED uses a class balancing weight to make a balance between edge & non-edge region, to find the true & strong edges much effectively as shown in Fig. 3. To implement it, we have used the code provided at [4] for exploiting edge map predictions obtained through training the modified VGGNet with BSDS500 [50]. HED seems to produce better compression performance than CED as shown in Table I.

B. Comparison for learning the proposed network separately with VDSR & EDSR post-processing module

In [13], the VDSR module has been used as the post-processing module. However, we have used the EDSR module to remove the blocking effects generated after JPEG decoding. To show the efficacy of the EDSR module, we have implemented the proposed algorithm considering the VDSR post-processing module. The compression with the EDSR module achieves significant improvement over compression with the VDSR module as shown in Table II.

C. Comparison for impact of Pre & Post-processing module separately on codec compression with full framework

To check the individual contribution of each sub-module i.e., pre & post-processing module, we have trained the network

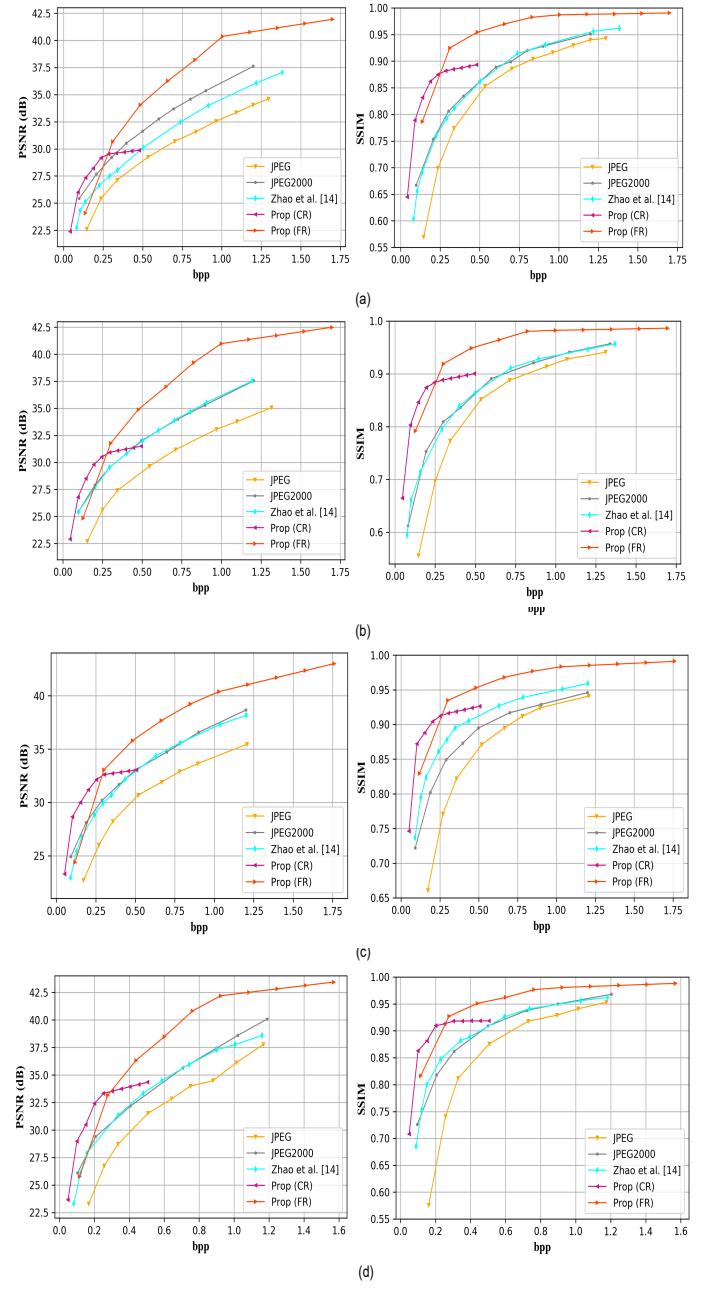[4] https://github.com/senliuy/Keras_HED_with_model

Fig. 7. Objective (PSNR) & Subjective parameter (SSIM) comparison of proposed algorithm with standard & downsampling method on (a) Live 1 (b) Set 14 (c) Set 7 (d) Set 5 datasets.

TABLE VII
COMPRESSION PERFORMANCE COMPARISON (PSNR(DB)/SSIM) OF PROPOSED ALGORITHM WITH JPEG & JIANG'S [13].

QF	5			10		
Algorithm Dataset	JPEG	Jiang's [13]	P_E (FR)	JPEG	Jiang's [13]	P_E (FR)
Set5	26.13/0.7206	29.20/0.8387	29.98/0.8561	28.99/0.8109	31.40/0.8854	31.67/0.9032
Set14	24.90/0.6686	26.89/0.7914	27.00/0.8423	27.49/0.7762	28.91/0.8336	30.21/0.8998
Live 1	24.60/0.6666	26.78/0.7934	26.93/0.8401	27.02/0.7720	28.84/0.8411	29.27/0.9008
General 100	25.93/0.7228	27.29/0.8447	28.10/0.8822	28.92/0.8199	30.16/0.8767	31.79/0.9324
Average	25.39/0.6947	27.54/0.8171	28.00/0.8552	28.11/0.7948	29.83/0.8592	30.74/0.9091

TABLE VIII
COMPARISON (PSNR(DB)/SSIM OF THE PROPOSED ALGORITHM WITH OTHER DEBLOCKING ALGORITHMS AT QF:10 AS REPORTED IN [13] ON SET 7
DATASET (RESULTS OF OTHER METHODS HAVE BEEN TAKEN FROM JIANG'S PAPER [13]).

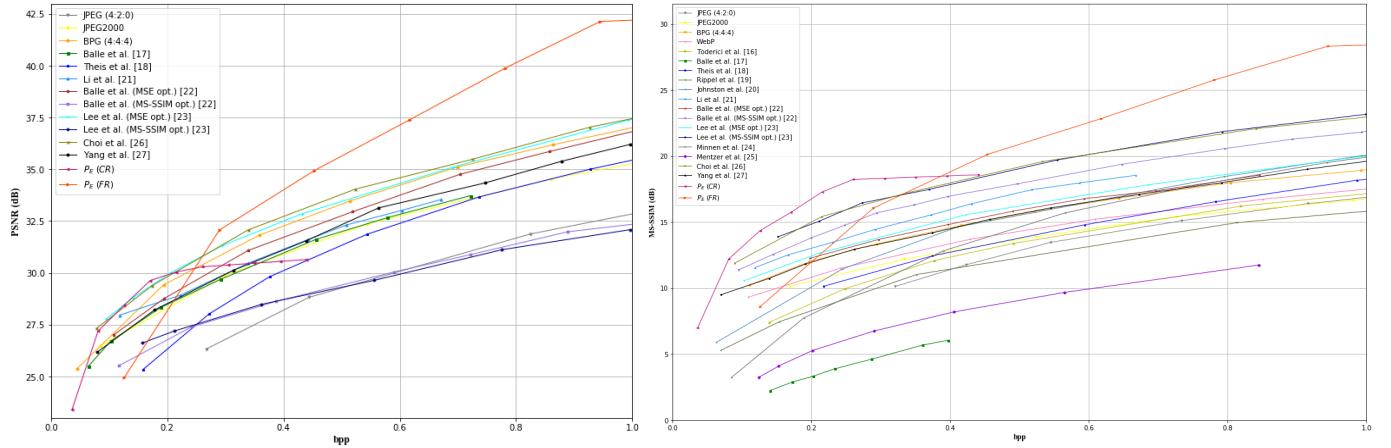| Test Images	Lena	Butterfly	Cameraman	House	Peppers	Parrots	Leaves	Average
JPEG [1]	25.24/0.8325	26.47/0.7965	30.56/0.8183	30.41/0.8183	30.14/0.7839	25.40/0.8609	28.96/0.8336	28.17/0.8206
Sun's [3]	26.52/0.8871	27.26/0.8358	32.00/0.8504	31.72/0.8590	31.62/0.8322	26.60/0.9138	30.04/0.8783	29.39/0.8652
Foi's [4]	31.84/0.8586	27.25/0.9014	27.48/0.8333	32.09/0.8491	31.69/0.8318	30.15/0.8778	27.30/0.9282	29.69/0.8686
BM3D [5]	26.64/0.8896	27.25/0.8240	32.07/0.8492	31.77/0.8549	31.42/0.8250	26.98/0.9207	30.05/0.8749	29.45/0.8626
Zhang's [6]	26.83/0.8923	27.45/0.8329	32.11/0.8513	31.92/0.8597	31.68/0.8317	27.26/0.9212	30.50/0.8804	29.68/0.8671
Ren's [7]	27.17/0.9010	27.43/0.8259	32.41/0.8526	31.92/0.8571	31.63/0.8300	27.59/0.9309	30.34/0.8775	29.78/0.8679
DicTV [8]	26.09/0.8699	26.92/0.8046	31.77/0.8484	31.55/0.8559	31.29/0.8244	26.33/0.9032	29.82/0.8741	29.11/0.8544
WNNM [9]	27.22/0.9019	27.40/0.8248	32.42/0.8531	31.93/0.8571	31.64/0.8303	27.66/0.9325	30.33/0.8755	29.80/0.8681
CONCOLOR [10]	27.09/0.9142	27.72/0.8401	33.04/0.8609	32.19/0.8661	31.94/0.8358	28.20/0.9406	30.66/0.8842	30.12/0.8774
ARCNN [12]	28.54/0.9237	27.62/0.8389	32.53/0.8591	32.0/0.8711	31.50/0.8434	28.31/0.9495	30.62/0.8942	30.16/0.8828
ComCNN [13]	26.32/0.8796	27.05/0.8154	31.82/0.8497	31.63/0.8573	31.04/0.8296	26.51/0.9095	29.97/0.8766	29.12/0.8597
RecCNN [13]	28.04/0.9192	27.33/0.8429	32.76/0.8584	32.35/0.8679	31.34/0.8396	28.53/0.9443	30.85/0.8884	30.17/0.8801
Jiang's [13]	28.60/0.9245	27.44/0.8448	33.25/0.8678	33.11/0.8838	31.83/0.8532	28.77/0.9475	31.11/0.9047	30.59/0.8895
			FR					
P_rN	29.99/0.7938	25.37/0.8600	26.46/0.7956	30.50/0.8187	30.10/0.7834	31.83/0.8527	25.23/0.8245	28.50/0.8184
P_oN	31.19/0.8264	27.30/0.9190	27.65/0.8281	31.91/0.8465	31.47/0.8259	33.25/0.8912	27.53/0.8920	30.04/0.8613
P_E (FR)	32.28/0.8862	28.52/0.9598	29.13/0.9080	31.95/0.9078	31.97/0.8698	33.78/0.9361	29.06/0.9399	30.96/0.9154

Fig. 8. Objective (PSNR) & Subjective parameter (MS-SSIM) comparison of proposed algorithm with standard & recent state-of-the-art approaches on Kodak dataset. For showing comparison, $-10log_{10}(MS-SSIM)$ is plotted in dB to synchronize with the process adopted in recent state-of-the-art methods.

considering P_rN & P_oN individually. In fact, these individual implementations are the special cases of proposed P_E (FR). From Table III, it is inferred that traditional codec compression followed by the post-processing network P_oN produces better results than pre-processing module P_rN. Moreover, adding both these modules with the codec compression added further gain in compression performance.

D. Comparison for learning the network on Edge aware loss & comparison with the MSE loss function

The training of P_rN on edge-aware loss forced the network to retain structural and sharp details while compressing the images. Also, the tuning parameter $\alpha = 0.75$ worked best for training the network to preserve edges in the reconstructed images. The first term in Eq. (2) ensures high PSNR by minimizing MSE while preserving edges. Preserving edges via the second term in Eq. (2) ensures the preservation of structural information and hence maximize SSIM & MS-SSIM. High mIoU value in Table IV indicate to retain edges with edge loss
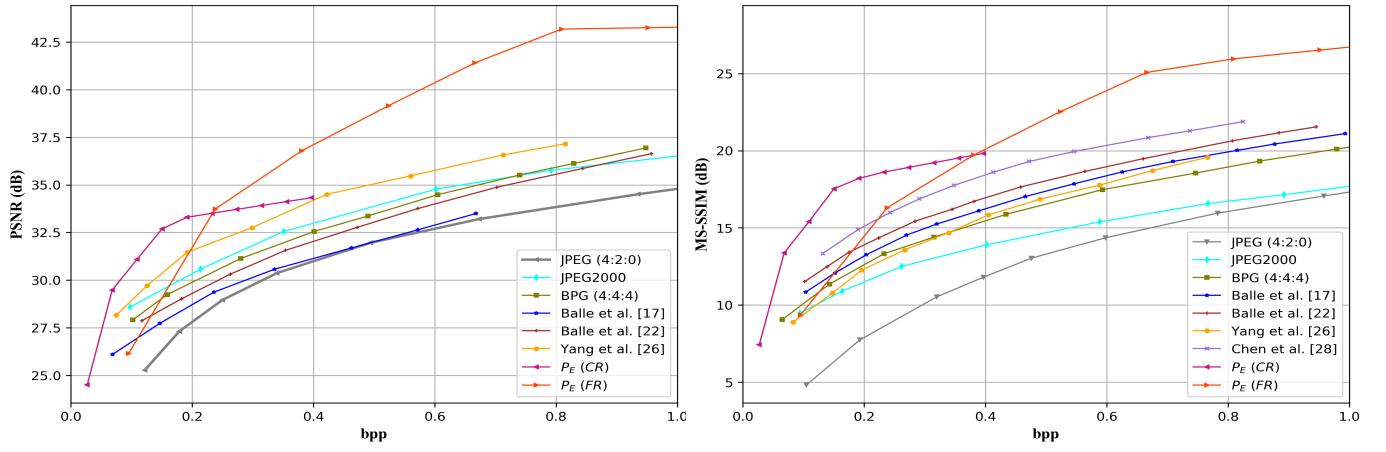
Fig. 9. Objective (PSNR) & Subjective parameter (MS-SSIM) comparison of proposed algorithm with standard & recent state-of-the-art approaches on CLIC 2019 dataset.

TABLE IX
PSNR(DB)/SSIM COMPARISON OF VARIOUS DEBLOCKING &
POST-PROCESSING METHODS WITH THE PROPOSED ALGORITHM ON
CLASSIC 5 DATASET.

QF Algorithm	10	20	30	40
DnCNN-3 [11]	29.40/0.8026	31.63/0.8610	32.90/0.8860	33.77/0.9003
ARCNN [12]	29.05/0.7929	31.16/0.8517	32.52/0.8806	33.33/0.8953
DPW-SDNet [44]	29.74/0.8124	31.95/0.8663	33.22/0.8903	34.07/0.9039
P_E (FR)	30.18/0.8817	32.26/0.9148	33.90/0.9357	34.59/0.9405

TABLE X
PSNR(DB)/SSIM COMPARISON OF VARIOUS DEBLOCKING &
POST-PROCESSING METHODS WITH THE PROPOSED ALGORITHM ON LIVE
1 DATASET.

QF Algorithm	10	20	30	40
DnCNN-3 [11]	29.19/0.8123	31.59/0.8802	32.98/0.9090	33.96/0.9247
ARCNN [12]	28.98/0.8217	31.29/0.8871	32.69/0.9166	33.63/0.9306
DPW-SDNet [44]	29.53/0.8210	31.90/0.8854	33.31/0.9130	34.30/0.9282
CASCNN [45][3]	29.44/0.8330	31.70/0.8950	-	34.10/0.9370
Galteri's [46]	29.45/0.8340	31.77/0.8960	33.15/0.9220	34.09/0.9350
P_E (FR)	29.27/0.9008	30.90/0.9288	32.74/0.9475	33.93/0.9535

TABLE XI
PSNRB(DB) COMPARISION OF VARIOUS DEBLOCKING &
POST-PROCESSING METHODS WITH THE PROPOSED ALGORITHM ON LIVE
1 DATASET.

QF	10	20	30	40
ARCNN [12]	28.77	30.79	32.22	33.14
CASCNN [45]	29.19	30.88	-	33.68
DnCNN-3 [11]	28.91	31.08	32.35	33.29
Galteri's [46]	29.10	31.26	32.51	33.40
DPW-SDNet [44]	29.13	31.27	32.52	33.44
P_E (FR)	29.27	30.89	32.74	33.92

TABLE XII
PSNRB(DB) COMPARISION OF VARIOUS DEBLOCKING &
POST-PROCESSING METHODS WITH THE PROPOSED ALGORITHM ON
CLASSIC 5 DATASET.

QF	10	20	30	40
ARCNN [12]	28.78	30.60	32	32.81
DnCNN-3 [11]	29.13	31.19	32.26	33.20
DPW-SDNet [44]	29.37	31.42	32.51	33.24
P_E (FR)	30.14	32.21	33.83	34.51

TABLE XIII
BD-RATE (IN %)/BD-PSNR (DB) FOR CODING GAIN OBTAINED
COMPARING PROPOSED ALGORITHM WITH-RECENT STATE-OF-THE ART
TECHNIQUES.

Other Algo \ Proposed	P_E (CR)	P_E (FR)
Set 5		
JPEG	-68.04/5.71	-56.67/5.96
JPEG2000	-44.99/2.67	-33.54/2.71
Zhao's [14]	-44.76/2.97	-33.72/2.85
Set 7		
JPEG	-69.81/5.43	-58.98/6.01
JPEG2000	-42.96/2.30	-30.76/2.39
Zhao's [14]	-48.12/2.94	-33.94/2.73
Set 14		
JPEG	-66.59/4.45	-57.15/5.70
JPEG2000	-33.72/1.41	-29.00/2.35
Zhao's [14]	-35.25/1.51	-29.50/2.39
Live 1		
JPEG	-61.63/3.22	-49.80/4.75
JPEG2000	-20.70/0.4997	-20.16/1.61
Zhao's [14]	-43.84/1.96	-36.47/3.20
Kodak		
JPEG	-69.44/3.07	-64.18/6.97
JPEG2000	-29.87/0.93	-42.00/3.00
BPG	-3.98/-0.12	-14.85/1.65
Balle's [17]	-30.69/0.95	-13.33/1.88
Theis's [18]	-57.74/2.31	-43.94/4.16
Balle's (MSE opt.) [22]	-25.83/0.49	-26.05/1.97
Balle's (MS-SSIM opt.) [22]	-62.19/2.77	-57.87/5.43
Lee's (MSE opt.) [23]	2.52/-0.43	-19.85/1.36
Lee's (MS-SSIM opt.) [23]	-63.97/2.55	-61.10/5.88
Choi's [26]	3.44/-0.42	-16.89/1.28
CLIC 2019		
JPEG2000	-71.26/4.51	-54.22/5.07
BPG	-59.19/3.14	-44.14/3.55
Balle's [17]	-68.87/4.23	-44.36/4.24
Balle's [22]	-64.51/3.70	-49.74/4.33
Yang's [27]	-39.78/1.72	-22.09/1.81

(in P_E) as compared to MSE loss function (in P_M), ultimately ensuring the edge-preserving property. Also, it indicates that an FR network can recover images with quite a higher mIoU than a CR representation network. In addition to this, we have trained both the modules with MSE and the results are given in Table V. The results show that training of P_rN with

edge-aware loss function leads to better results than training with MSE. Apart from that, Li et al. [15] use a resolution decrement parameter (layer with stride 2) at the first CNN layer. But, we believe it should be placed at the last CNN layer (as used in the proposed algorithm) because gradual or delayed downsampling help in learning more features as compared to sudden downsampling just after the first layer. This sudden downsampling may lose learning of some features which are crucially required. To support this statement, a small experiment is conducted first by taking stride 2 in the first layer, then in last and the results are shown in Table VI. With multiple sets of experiments, we finalize the training of P_rN & P_oN module on edge-aware & MSE loss functions, respectively. It was found that training with edge-aware loss functions performed well with comparatively more bit saving.

E. Comparison of CR learning framework with FR learning framework

The results in Fig. 4 stressed upon the point of more appropriate learning of features showing the superiority of the FR over CR image representation in terms of compression gain and performance measures. From the highlighted region in the image given in Fig. 5, it is quite evident that at low bit-rates, the proposed P_E (CR) & P_E (FR) reduces the blocking artifacts more effectively compared to JPEG, due to super-resolution network introduced to post-process the image. However, it is noted that better performance at low bit-rates using CR network representation is obtained at the cost of performance saturation at high bit-rates which can be seen from Fig. 6.

V. COMPARISON WITH STATE-OF-THE-ART ALGORITHMS

We have compared the proposed algorithm with JPEG and Jiang's [13] for QF of 5 & 10 on Set 5, Set 14, Live 1 & General 100 dataset. It is observed that the proposed P_E (FR) has outperformed the mentioned approaches with a significant margin as shown in Table VII. Specifically, we have compared the proposed algorithm with various deblocking and post-processing based algorithms and again found P_E (FR) to overcome the approaches shown in Table VIII. Here also, the contribution of each module P_rN & P_oN can be seen as demonstrated separately.

When tested on Classic 5 & Live 1 dataset, the proposed algorithm clearly outperformed the work reported in [11], [12] as shown in Table IX & Table X respectively. The proposed algorithm is effectively seen as to remove blocking artifacts which are confirmed when comparing PSNRB with standard artifacts reduction techniques as shown in Table XI & Table XII.

When tested with Live 1, Set 14, Set 7 & Set 5 dataset, the proposed algorithm has effectively outperformed JPEG, JPEG2000 & Zhao's [14] as shown in Fig. 7). For plotting rate-distortion curves, the bit-rate for a file is obtained by dividing the file size of codec compressed bit-stream with the respective original image dimension. With the proposed algorithm, the improvement in PSNR is approximately 25%, 3.77%, 10%, and 22.73%, 12.5%, 19.12% at low and high bit-rates respectively, compared to JPEG, JPEG2000, and Jiang's

algorithms on Live 1 dataset. This improvement in SSIM is approximately 45.6%, 15.28%, 16.9%, and 10.11%, 6.5%, 6.52% at low and high bit-rates respectively, compared with the same approaches. When tested on the Set 14 dataset, the improvement in PSNR is 26.67%, 9.61%, 9.61%, and 20.31%, 13.24%, 11.59% at low and high bit-rates respectively, compared to JPEG, JPEG2000 and Jiang's algorithms. Similarly, the increase in SSIM is approximately 41.67%, 16.44%, 19.72%, and 11.90%, 6.67%, 5.49% at low and high bit-rates respectively, compared with the same approaches. On the Set 7 dataset, the improvement in PSNR is approximately 30.43%, 11.11%, 15.38%, and 16.92%, 7.04%, 8.57% at low and high bit-rates respectively, compared to JPEG, JPEG2000 and Jiang's algorithms. Similarly, the improvement in SSIM is 31.82%, 12.99%, 8.75%, and 6.59%, 5.43%, 3.19% at low and high bit-rates respectively, compared with the same approaches. The proposed algorithm outperformed JPEG, JPEG2000, and Jiang's with PSNR with an approximate gain of 38.30%, 12.07%, 16.07%, and 19.40%, 12.68%, 12.68% at low and high bit-rates respectively, on Set 5 dataset. The improvement in SSIM is approximately 42.59%, 3.90%, 7.3%, and 7.87%, 3.23%, 2.13% at low and high bit-rates respectively, compared with the same approaches. As shown in Fig. 8, when tested on Kodak benchmark dataset, the proposed algorithm outperformed JPEG, JPEG2000, BPG, and the work reported in [17], [18], [21], [22], [23], [26], [27] in terms of PSNR by an approximate improvement of 20.75%, 8.47%, 3.22%, 8.47%, 14.29%, 4.17%, 4.92%, 3.23%, 3.23%, 5.26% and 24.59%, 14.46%, 10.14%, 13.43%, 16%, 13.64%, 11.76%, 8.57%, 8.57%, 14.94% at low and high bit-rates respectively. Similarly the improvement in MS-SSIM is approximately 71.43%, 50%, 36.36%, 23.08%, 56.52%, 40.63%, 63.64%, 50%, 70%, 25.93%, 9.09%, 6.5%, 13.04%, 67%, 64.70%, 41.67% and 64.47%, 61.29%, 47.06%, 51.52%, 60.26%, 43.68%, 78.57%, 51.52%, 32.35%, 27.78%, 21.95%, 35.14%, 42.86%, 69%, 16.28%, 35.29% at low and high bit-rates respectively, when compared with JPEG, JPEG2000, BPG, WebP algorithms and the work in [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27].

When tested on CLIC 2019 challenging dataset as shown in Fig. 9, the proposed algorithm outperformed JPEG, JPEG2000, BPG and the work in [17], [22], [27] by approximately 20.73%, 8.85%, 10.53%, 16.24%, 11.70%, 6.78% and 23.08%, 14.94%, 17.39%, 22.73%, 19.12%, 14.08% at low and high bit-rates respectively. Similarly the improvement in MS-SSIM is approximately 72%, 45.45%, 39.13%, 33.33%, 23.08%, 52.38%, 18.52% and 71.43%, 50%, 41.18%, 33.33%, 29.73%, 37.14%, 17.07% at low and high bit-rates respectively, when compared to JPEG, JPEG2000, BPG and the work in [17], [22], [27], [28]. More gain in terms of objective (PSNR) & subjective (MS-SSIM) evaluation has been achieved at lower as well as higher bit-rates due to the use of a super-resolution network that uses residual learning. The edge-aware loss function also helped to reduce the bit-rate while maintaining the same quality of decompressed images. The coding gain of the proposed algorithm (BD metric calculation) concerning various state-of-the-art methods is also found to be quite high as given in Table XIII. High

TABLE XIV
SIMILARITY TESTS (AVERAGE VALUE) BETWEEN TRAINING SET (BSD400) AND TESTING SETS USED.

Datasets	Set 5	General 100	Set 7	CLIC 2019	Set 14	Kodak	Live 1	Classic 5
Correlation Coefficient	0.0488	0.0985	0.109	0.1106	0.1164	0.126	0.1521	0.2475

TABLE XV
RUN TIME COMPLEXITY (IN SEC) OF PROPOSED ALGORITHM WITH VARIOUS STATE-OF-THE-ART ALGORITHMS FOR COMPRESSION OF 256×256 SIZE IMAGE AT QF:10.

Algorithm	Time (CPU/GPU)	Algorithm	Time (CPU/GPU)
ARCNN [12]	-/0.33	Zhang's [10]	442/-
Jiang's [13]	1.56/0.017	DnCNN-3 [11]	-/0.0157
Sun's [3]	132/-	Rippel's [19]	-/0.01
BM3D [5]	3.40/-	Balle's [22]	0.7/-
Zhang's [6]	251/-	Lee's [23]	-/7.5
Ren's [7]	36/-	DPW-SDNet [44]	-/1.2 ∼ 2
DicTV [8]	53/-	CASCNN [45]	-/2.4
WNNM [9]	230/-	*Proposed CR/FR*	-/0.081

TABLE XVI
TRAINING PARAMETERS OR MODEL SIZE COMPARISON OF PROPOSED ALGORITHM WITH VARIOUS STATE-OF-THE-ART METHODS.

Algorithm	Parameters	Algorithm	Parameters
ARCNN [12]	106564 (417 KB)	Li et al. [21]	257557187
Jiang's [13]	708674	Balle's [22]	4.76 million
Zhao's [14]	19968768	Mentzer's [25]	9561472
DnCNN-3 [11]	1112192	Yang's [27]	10.27 million
Toderici's [16]	7457472	Chen's [28]	261.803 MB
Balle's [17]	16.1 million	DPW-SDNet [44]	1700736
Theis's [18]	2742112	CASCNN [45]	5144000
Johnston's [20]	9917504	Galteri's [46]	7128160
Proposed CR/FR	796098/648386		

coding gain means that a very large amount of bit-saving takes place on the application of proposed compression-decompression algorithm. Comprehensively, it can be deduced that the objective (PSNR) and perceptual subjective quality metric (SSIM & MS-SSIM), shows significant improvement with compression rates. During the investigation of the results obtained, we have also conducted a correlation analysis to check the similarity/dissimilarity between training and testing sets in Table XIV. For this, we have calculated the correlation coefficient as a metric, which seems to synchronize with the compression performance obtained in rate-distortion curves. From Fig. 7, Fig. 8, Fig. 9 & Table XIV, it is clear that if the correlation coefficient between training and all testing sets is more, better compression performance will be obtained on that test set & vice-versa. Through the extensive investigation, we report that, $P_E\ (CR)$ & $P_E\ (FR)$ representation network is suitable for low & high bit-rates respectively. The compression performance is saturated at high bit-rates when using the CR representation network. Since it is a well-known fact that to display images on Ultra high definition (UHD), initial downsampling of images is mandatory. So compact-resolution image compression $P_E\ (CR)$ can be preferred where transmission bandwidth is limited & vice-versa. Furthermore, training with a large size dataset did not help in improving the efficiency of the compression algorithm, which is also supported by [13]. Compared with standard approaches, deblocking methods, post-processing methods, recently reported state-of-

the-art techniques, the approach yields very good results with only a few training images. The testing time when using the proposed algorithm is also less i.e., 0.081s, making it practically implementable. Table XV and Table XVI show the comparison between the run-time complexity and number of training parameters for the proposed algorithm. Both the testing time and training parameters being feasible to make the proposed algorithm real-time applicable.

VI. CONCLUSION

The end-to-end compression-decompression framework centered on super-resolution preserves the quality of the images both at low & high bit-rates along with reducing artifacts while reconstructing the images. The edge-aware loss function introduced using HED is used to prevent the blurriness & artifacts which are generally caused by training the network with MSE. We reported on extensive experiments conducted to evaluate the proposed approach on several compression datasets. The results demonstrate that compared to the various state-of-the-art method, both the super-resolution post-processing network & the edge-aware loss function provide significant gain concerning deblocking, rate-distortion performance, compression performance (PSNR, PSNRB, SSIM, MS-SSIM), and coding gain in terms of BD-Rate & BD-PSNR.

REFERENCES

[1] ISO/IEC 10918-1:1994, "Digital Compression and Coding of Continuous-tone Still images: Requirements and Guidelines," 1994.

[2] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. on Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, 2000.

[3] D. Sun and W.-K. Cham, "Postprocessing of Low Bit-Rate Block DCT Coded Images Based on a Fields of Experts Prior," *IEEE Trans. on Image Proc.*, vol. 16, no. 11, pp. 2743–2751, 2007.

[4] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images," *IEEE Trans. on Image Proc.*, vol. 16, no. 5, pp. 1395–1411, 2007.

[5] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian, "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," *IEEE Trans. on Image Proc.*, vol. 16, pp. 2080–2095, 2007.

[6] X. Zhang, R. Xiong, X. Fan, S. Ma, and W. Gao, "Compression Artifact Reduction by Overlapped-Block Transform Coefficient Estimation With Block Similarity," *IEEE Trans. on Image Proc.*, vol. 22, no. 12, pp. 4613–4626, 2013.

[7] J. Ren, J. Liu, M. Li, W. Bai, and Z. Guo, "Image Blocking Artifacts Reduction via Patch Clustering and Low-Rank Minimization," in *2013 Data Compression Conf.* IEEE, 2013, pp. 516–516.

[8] H. Chang, M. K. Ng, and T. Zeng, "Reducing Artifacts in JPEG Decompression Via a Learned Dictionary," *IEEE Trans. on Signal Proc.*, vol. 62, no. 3, pp. 718–728, 2013.

[9] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted Nuclear Norm Minimization with Application to Image Denoising," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 2862–2869.

[10] J. Zhang, R. Xiong, C. Zhao, Y. Zhang, S. Ma, and W. Gao, "CON-COLOR: Constrained Non-Convex Low-Rank Model for Image Deblocking," *IEEE Trans. on Image Proc.*, vol. 25, no. 3, pp. 1246–1259, 2016.

[11] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Trans. on Image Proc.*, vol. 26, no. 7, pp. 3142–3155, 2017.

[12] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression Artifacts Reduction by a Deep Convolutional Network," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2015, pp. 576–584.

[13] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, "An End-to-End Compression Framework Based on Convolutional Neural Networks," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 3007–3018, 2018.

[14] L. Zhao, H. Bai, A. Wang, and Y. Zhao, "Learning a virtual codec based on deep convolutional neural network to compress image," *Journal of Visual Comm. and Image Rep.*, vol. 63, pp. 102 589.1–11, 2019.

[15] Y. Li, D. Liu, H. Li, L. Li, Z. Li, and F. Wu, "Learning a convolutional neural network for image compact-resolution," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1092–1107, 2018.

[16] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full Resolution Image Compression with Recurrent Neural Networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.

[17] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end Optimized Image Compression," *Int. Conf. on Learning Representations*, 2017.

[18] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy Image Compression with Compressive Autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.

[19] O. Rippel and L. Bourdev, "Real-Time Adaptive Image Compression," in *Proc. of the 34th Int. Conf. on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 2922–2930.

[20] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, and G. Toderici, "Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4385–4393.

[21] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning Convolutional Networks for Content-Weighted Image Compression," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 3214–3223.

[22] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.

[23] J. Lee, S. Cho, and S.-K. Beack, "Context-adaptive Entropy Model for End-to-end Optimized Image Compression," *arXiv preprint arXiv:1809.10452*, 2018.

[24] D. Minnen, J. Ballé, and G. D. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," in *Advances in Neural Information Proc. Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 10 771–10 780.

[25] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Conditional Probability Models for Deep Image Compression," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4394–4402.

[26] Y. Choi, M. El-Khamy, and J. Lee, "Variable Rate Deep Image Compression with a Conditional Autoencoder," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2019, pp. 3146–3154.

[27] F. Yang, L. Herranz, J. van de Weijer, J. A. I. Guitián, A. M. López, and M. G. Mozerov, "Variable Rate Deep Image Compression With Modulated Autoencoder," *IEEE Signal Proc. Letters*, vol. 27, pp. 331–335, 2020.

[28] T. Chen and Z. Ma, "Variable Bitrate Image Compression with Quality Scaling Factors," in *ICASSP 2020 - 2020 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2020, pp. 2163–2167.

[29] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshop*, 2017, pp. 136–144.

[30] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *Proc. of the IEEE Conf. on Computer vision and Pattern Recognition*, 2016, pp. 1646–1654.

[31] R. K. Pandey, N. Saha, S. Karmakar, and A. Ramakrishnan, "MSCE: An Edge-Preserving Robust Loss Function for Improving Super-Resolution Algorithms," in *Int. Conf. on Neural Information Proc.* Springer, 2018, pp. 566–575.

[32] G. Seif and D. Androutsos, "Edge-Based Loss Function for Single Image Super-Resolution," in *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2018, pp. 1468–1472.

[33] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1132–1140.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *Int. Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[35] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "LIVE Image Quality Assessment Database Release 2 (2005)," 2005.

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

[37] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," *BMVA press*, 2012.

[38] C. B. MacKnight, "Kodak Photo CD Eastman Kodak Company Kodak Information Center Department E 343 State Street Rochester, NY 14650-0811," *Journal of Computing in Higher Education*, vol. 7, no. 1, pp. 129–131, Sep 1995. [Online]. Available: https://doi.org/10.1007/BF02946148

[39] Workshop And Challenge On Learned Image Compression, "CLIC 2019 dataset," 2019. [Online]. Available: http://www.compression.cc/challenge/

[40] "Classic 5 dataset," accessed December 2019. [Online]. Available: https://github.com/cszn/DnCNN/tree/master/testsets/classic5

[41] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European Conf. on Computer Vision.* Springer, 2016, pp. 391–407.

[42] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014.

[43] D. Mishra, S. K. Singh, and R. K. Singh, "Wavelet-based deep auto encoder-decoder (wdaed)-based image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 4, pp. 1452–1462, 2021.

[44] 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 *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 711–720.

[45] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A Deep Convolutional Neural Network for Image Compression Artifact Suppression," in *2017 Int. Joint Conf. on Neural Networks (IJCNN).* IEEE, 2017, pp. 752–759.

[46] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo, "Deep Generative Adversarial Compression Artifact Removal," in *Proc. of the IEEE Int. Conf. on Computer Vision*, 2017, pp. 4826–4835.

[47] C. Yim and A. C. Bovik, "Quality Assessment of Deblocked Images," *IEEE Trans. on Image Proc.*, vol. 20, no. 1, pp. 88–98, 2010.

[48] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *VCEG-M33*, 2001.

[49] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[50] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.