Better Compression With Deep Pre-Editing

Hossein Talebi, Damien Kelly, Xiyang Luo, Ignacio Garcia Dorado, Feng Yang, *Senior Member, IEEE*, Peyman Milanfar, *Fellow, IEEE*, and Michael Elad, *Fellow, IEEE*

***Abstract*— Could we compress images via standard codecs while avoiding visible artifacts? The answer is obvious – this is doable as long as the bit budget is generous enough. What if the allocated bit-rate for compression is insufficient? Then unfortunately, artifacts are a fact of life. Many attempts were made over the years to fight this phenomenon, with various degrees of success. In this work we aim to break the unholy connection between bit-rate and image quality, and propose a way to circumvent compression artifacts by pre-editing the incoming image and modifying its content to fit the given bits. We design this editing operation as a learned convolutional neural network, and formulate an optimization problem for its training. Our loss takes into account a proximity between the original image and the edited one, a bit-budget penalty over the proposed image, and a no-reference image quality measure for forcing the outcome to be visually pleasing. The proposed approach is demonstrated on the popular JPEG compression, showing savings in bits and/or improvements in visual quality, obtained with intricate editing effects.**

***Index Terms*— Image compression, image enhancement, pre-filtering.**

I. Introduction

COMMONLY used still image compression algorithms, such as JPEG [55], JPEG-2000 [15], HEIF [1] and WebP [25] produce undesired artifacts when the allocated bit rate is relatively low. Blockiness, ringing, and other forms of distortion are often seen in compressed-decompressed images, even at intermediate bit-rates. As such, the output images from such a compression procedure are of poor quality, which may hinder their use in some applications, or more commonly, simply introduce annoying visual flaws.

Numerous methods have been developed over the years to confront this problem. In Section IV we provide a brief review of the relevant literature, encompassing the various strategies taken to fight compression artifacts. Most of the existing solutions consider a post-processing stage that removes such artifacts after decompression [4], [5], [14], [16], [20], [21], [33], [34], [37], [45], [51], [57], [60]–[62]. Indeed, hundreds of papers that take this post-processing approach have been published over the years, including recent deep-learning based solutions (e.g., [13], [19], [23], [56]).

Manuscript received February 28, 2020; revised July 8, 2020 and February 21, 2021; accepted June 20, 2021. Date of publication July 15, 2021; date of current version July 26, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jing-Ming Guo. *(Corresponding author: Hossein Talebi.)*

The authors are with Google Research, Mountain View, CA 94043 USA (e-mail: htalebi@google.com).

Digital Object Identifier 10.1109/TIP.2021.3096085

Far less popular are algorithms that propose to pre-process the image prior to its compression, in order to reduce its entropy, thus avoiding the creation of artifacts in the first place [17], [18], [36], [43], [44], [53]. Indeed, a denoising applied before the compression is often found effective for better encoding performance (e.g. [48]). This line of thinking is scarcer in the literature due to the more complex treatment it induces and the weaker control it provides on the output artifacts. Still, such a pre-processing approach has a great advantage over the alternatives, as the changes to the image are done on the server side, while the decoder side does not need to be modified nor adjusted.

In this work we propose to pre-process the image by automatically editing its content, applied before its compression using JPEG standard. Our goal is to modify the image content smartly so as to guarantee that (i) most of the visual information in the image is preserved; (ii) the subsequent compression operates in a much better regime and thus leads to reduced artifacts; and (iii) the edited image after compression is still visually appealing. By considering all these forces holistically, we aim to get creative editing effects that enable the compression-decompression stage to perform at its best for the given bit budget.

While one could pose the proposed editing task as an optimization problem to be solved for each incoming image separately, we take a more challenging route, in which we target the design of a universal deep neural network that performs the required editing on an arbitrary input image. The clear advantage in this approach is the speed with which inference is obtained once the network has been trained.

Our learning relies on minimizing a loss-function that includes three key penalties, aligned with the above description. The first forces the original and the edited images to be "sufficiently close" to each other, while still allowing content editing. A second term penalizes the bit content of the edited image, so as to force the bit-budget constraint while striving for an artifact-free compression. This part is achieved by yet another network [8] that predicts the entropy and quality of the image to be compressed. Last, but definitely not least, is a third penalty that encourages the edited image after compression to be visually pleasing. Indeed, our formulation of the problem relies heavily on the availability of a no-reference quality metric, a topic that has seen much progress in recent years [11], [30], [40]–[42], [50], [59]. All the above-mentioned ingredients are posed as differentiable machines, enabling an end-to-end effective learning of the editing operation. An example of the proposed technique is shown in Fig. 1,

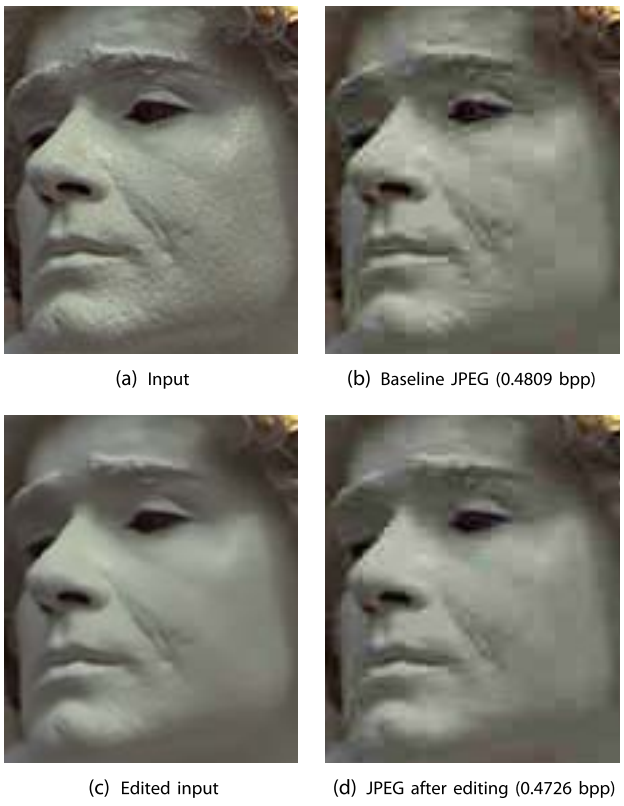


Fig. 1. Comparison of our pre-editing method with baseline JPEG. The uncompressed input (a) is compressed by JPEG (b), which shows a lot of compression artifacts. We propose to edit the input image (c) before JPEG compression (d) to obtain a better perceptual quality and lower bit rate.

where the editing operation allows for better perceptual quality and lower bit budget.

II. Formulating the Problem

We start with a few definitions that will help in formulating our problem and its solution.

Definition 1 (Codec Operation): We define by $C_R(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^N$ the process of compression and decompression of a given image with R bits. This function gets an image $\mathbf{x}$ and produces an image, $C_R(\mathbf{x})$, possibly with the compression artifacts mentioned above.

Definition 2 (Quality Assessment): We define by $Q(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^+$ the process of allocating a no-reference quality to a given image $\mathbf{x}$. The output is a non-negative scalar with values tending to zero for higher quality images.

Definition 3 (Distance Measure): We define by $dist(\mathbf{x}_1, \mathbf{x}_2)$ the distance between two images, $\mathbf{x}_1$ and $\mathbf{x}_2$, of the same size. Our distance function should be defined such that it is "forgiving" to minor content changes such as small geometrical shifts or warps, delicate variations in gray-values, or removal of fine texture.

Armed with the above, we are now ready to formulate our problem. Given an image $\mathbf{z}$ to be compressed with a bit budget of R bits, the common practice is to perform compression and decompression directly, $\hat{\mathbf{z}} = C_R(\mathbf{z})$, and live with the limitations.

In this work we suggest a novel alternative: We seek a new image $\mathbf{x}$ that is (i) as close as possible to the given image $\mathbf{z}$; (ii) it is compressible using R bits; and most importantly (iii) it is of high quality. Naturally, $\mathbf{x}$ will be an edited variation of $\mathbf{z}$ in which some of the content has been changed, so as to enable good quality despite the compression. Here is our first attempt to formulate this problem:

$$\min_{\mathbf{x}} \; dist(\mathbf{x}, \mathbf{z}) + \lambda Q(\mathbf{x}) \quad s.t. \; \mathbf{x} = C_R(\mathbf{x}). \tag{1}$$

In words, given $\mathbf{z}$ and R we seek an image $\mathbf{x}$ that is close to $\mathbf{z}$, it is of high quality (low value of $Q(\mathbf{x})$), and it can be represented via R bits. Referring to the constraint, recall that the compression-decompression operation is idempotent, i.e. applying it more than once on a given image results with the same outcome as using it once [32]. Thus, the constraint aims to say that $\mathbf{x}$ is a feasible outcome of the compression algorithm with the given budget of R bits.

An alternative formulation that may serve the same goal is one in which we fix the quality as a constraint as well,

$$\min_{\mathbf{x}} \; dist(\mathbf{x}, \mathbf{z}) \quad s.t. \; \mathbf{x} = C_R(\mathbf{x}) \text{ and } Q(\mathbf{x}) = Q_0, \tag{2}$$

so as to say that whatever happens, we insist on a specific output quality, willing to sacrifice content accordingly.

Both problems defined in Equations (1) and (2), while clearly communicating our goal, are hard to handle. This is mainly due to the non-differentiable nature of the function $C_R(\mathbf{x})$, and the fact that it is hard to fix a rate R while modifying the image $\mathbf{x}$. While these could be dealt with by a projection point of view (see [9]), we take a different route and modify our formulation to alleviate these difficulties. This brings us to the following additional definitions:

Definition 4 (Quality-Driven Codec Operation): We define by $C_q(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^N$ the process of compression and decompression of a given image with a quantization (or quality factor) q. This function gets an image $\mathbf{x}$ and produces an image, $C_q(\mathbf{x})$, possibly with the compression artifacts mentioned above.

Definition 5 (Entropy Predictor): We define by $B_q(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^+$ the process of predicting the compression-decompression performance of a specific algorithm (e.g., JPEG) for a given image $\mathbf{x}$ and a quantization level q. This function produces the expected file-size (or entropy).

Note that by setting q, we induce a roughly fixed PSNR on the image after compression-decompression. Thus, by minimizing $B_q(\mathbf{x})$ with respect to $\mathbf{x}$, we will be aiming for reducing the file size while preserving quality. Returning to our formulation, we add a penalty term, $B_q(\mathbf{x})$, so as to guarantee that the rate is preserved (or more accurately, controlled). This leads to

$$\min_{\mathbf{x}} \; dist(\mathbf{x}, \mathbf{z}) + \lambda Q(\mathbf{x}) + \mu B_q(\mathbf{x}) \quad s.t. \; \mathbf{x} = C_q(\mathbf{x}). \tag{3}$$

The constraint $\mathbf{x} = C_q(\mathbf{x})$ assures that $\mathbf{x}$ is a valid output of the compression, and it can be alternatively written as[1]

$$\min_{\mathbf{x}} \; dist(C_q(\mathbf{x}), \mathbf{z}) + \lambda Q(C_q(\mathbf{x})) + \mu B_q(C_q(\mathbf{x})). \tag{4}$$

[1] Admittedly, the notations introduced are a bit cumbersome, as both B and C use the same quantization level q. The alternative could have been to divide the compression C_q into an encoder and decoder, and feed the encoder result to B without specifying q. We chose to stay with the above formulation for consistency with the opening description.

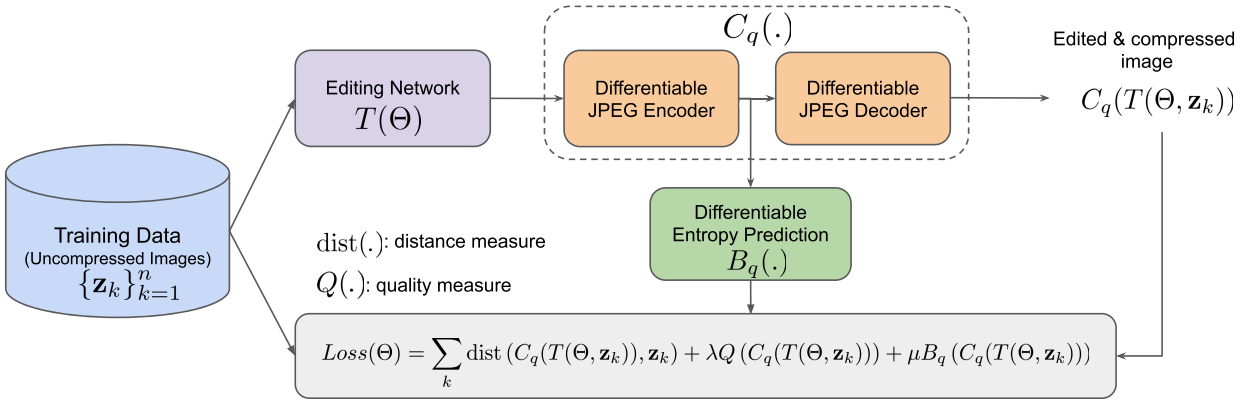


Fig. 2. Our learning pipeline for training the image editing $T(\Theta)$. Input image $\mathbf{z}_k$ is first edited by our editing network. Then, the edited image is fed to the differentiable JPEG encoder/decoder. The entropy of the quantized DCT coefficients are predicted and used in our training loss. To ensure that the compressed image is close to the uncompressed input, we use a distance measure. We also use a quality term to enforce the human perceptual preference.

If we have a differentiable proxy for the compression operation $C_q(\cdot)$, the above loss is manageable.

We could have stopped here, handling this optimization task and getting an edited and compressed image $\mathbf{x}$ for any incoming image $\mathbf{z}$. This could have been a worthy and even fascinating feat by itself, but we leave it for future work.

As we have already mentioned in the introduction, we aim higher. Our goal is to design a feed-forward CNN that would perform this editing for any given image automatically. Denoting this editing network by $T(\Theta, \mathbf{z})$, where Θ are the network parameters to be learned/set, our training loss is given by the following expression:

$$Loss(\Theta) = \sum_k \Big[dist\left(C_q(T(\Theta, \mathbf{z}_k)), \mathbf{z}_k\right) + \lambda Q\left(C_q(T(\Theta, \mathbf{z}_k))\right) + \mu B_q\left(C_q(T(\Theta, \mathbf{z}_k))\right) \Big]. \quad (5)$$

This expression simply sums the per-image loss over many training images $\{\mathbf{z}_k\}$, and replaces the edited image $\mathbf{x}_k$ by the network's output $T(\Theta, \mathbf{z}_k)$. Minimizing this loss with respect to Θ, we obtain the editing network, as desired. Our learning pipeline is shown in Fig. 2.

III. The Proposed Approach

In this section we dive into our implementation of the above-discussed editing idea. We start by specifying the ingredients used within the loss function and then turn to describe the training procedure employed.

A. Training Loss

Returning to our definitions, we would like to identify the specific ingredients used in Equation (5). In this work we concentrate on the JPEG compression algorithm, due to its massive popularity and the fact that it is central in digital imaging and mobile photography. Our formulation relies on three ingredients:

1) Distance Measure: We seek a definition of $dist(\mathbf{x}_1, \mathbf{x}_2)$ that does not penalize moderate editing changes between the two images. In our implementation we construct this distance measure by feature extraction function, $F(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^L$, and use the perceptual loss $\|F(\mathbf{x}_1) - F(\mathbf{x}_2)\|_1$ as our distance [24], [31], [54]. These features could be the activations of an inner layer within the VGG-16 [47] or the NIMA [50] networks, and they are used to characterize an image in a domain that is less sensitive to the allowed perturbations. The deeper these features are taken from, the more daring the editing of the image is expected to be. We experimented with various VGG-16 activations trained for image quality assessment task [50], and selected the output of the fifth convolutional block before max pooling as our feature extraction function $F(\mathbf{x})$.

2) Quality Measure: We assess image quality using NIMA [50]. NIMA is a no-reference image quality assessment machine that has been used for training image enhancement [49].

3) Differentiable JPEG: As mentioned above, we need to incorporate the function $C_q(\cdot)$ within our loss and thus it should be differentiable. Indeed, as we are working with JPEG, this function does not control the rate but rather the quality factor q when running this compression-decompression. The differentiable encoder/decoder consists of 4 steps:

1) Color conversion: We start with an RGB image, and convert to YUV color space. Since color conversion is basically a matrix multiplication, its derivatives are well defined for RGB to YUV and vice versa.
2) Chroma downsampling/upsampling: The YUV image may represent full chroma values (YUV444), or subsampled ones (YUV420). The downsampling operation to generate YUV420 is 2×2 average pooling. The upsampling is implemented with Bilinear interpolation.
3) DCT and inverse DCT: The DCT coefficients (D_{ij}) are computed for 8×8 image blocks of each YUV channel, separately. Note that the DCT and its inverse are matrix multiplications, and hence differentiable.
4) Quantization/Dequantization: Quantization is performed by 8×8 tables values U_{ij} as $\lfloor \frac{D_{ij}}{U_{ij}} \rceil$. Note that rounding to the nearest integer operation $\lfloor . \rceil$ has zero derivative

almost everywhere, and hence it does not work with our gradient-based learning framework. So, similar to [46] we employ a third-order polynomial approximation of the rounding operation as $\lfloor x \rceil + (x - \lfloor x \rceil)^3$ where x represents $\frac{D_{ij}}{U_{ij}}$.

Also, it is worth mentioning that the differentiable JPEG is only used at training, and all the test results presented in the paper use the standard JPEG to measure the bit-rate. So, the differentiable JPEG is only an approximation of the actual JPEG that allows us to train the pre-editing CNN.

4) Entropy Prediction: In our framework with JPEG, the discrete entropy of the quantized DCT coefficients should be measured. However, just as described above, the derivatives of the quantization operation are zero almost everywhere, and consequently gradient descent would be ineffective. To allow optimization via stochastic gradient descent, we use the entropy estimator proposed in [8]. While [8] represents an end-to-end compression scheme, our approach only borrows their entropy estimation technique. Based on this approach, we estimate the bit-rate consumed for JPEG compression by approximating the entropy of the quantized DCT coefficients. The approximated bit-rate can be expressed as $-\mathbb{E}[\log_2 \Pi_{\mathbf{d}}]$ where $\mathbf{d}$ represents the quantized DCT coefficients. As shown in [7], $\widehat{\mathbf{d}} = \mathbf{d} + \Delta\mathbf{d}$ is a continuous relaxation of the probability mass function $\Pi_{\mathbf{d}}$, where $\Delta\mathbf{d}$ is additive i.i.d. uniform noise with the same width as the quantization bins. This means that the differential entropy of $\widehat{\mathbf{d}}$ can be used as an approximation of $\mathbb{E}[\log_2 \Pi_{\mathbf{d}}]$.

We adopt the non-parametric approach of [8] to approximate $\Pi_{\widehat{\mathbf{d}}}$ marginals. Based on this method, a sigmoid function is used to approximate the cumulative density function of the DCT coefficients $\widehat{\mathbf{d}}$. The shift and the scale factors of the sigmoid are learned by a 4-layer neural network. Each intermediate layer of this network consists of 3 neurons followed by tanh activations. The density $\Pi_{\widehat{\mathbf{d}}}$ is computed as the derivative of the cumulative function. More implementation details are available in [6].

We train separate entropy estimators for each DCT coefficient set obtained from all 8×8 blocks: (i) Y channel DC (zero-frequency) coefficients, (ii) UV channels DC coefficients, (iii) Y channel non-DC coefficients, (iv) UV channels non-DC coefficients. This approach follows the actual JPEG encoder. The overall entropy is the sum of these four estimated entropies. We also follow the DPCM (Differential Pulse Code Modulation) framework to encode the difference between DC component of the DCT coefficients. Fig. 3 shows that the entropy estimator provides a close approximation of the actual JPEG bit-rates. Data points in Fig. 3 correspond to a total of 72 images compressed with various JPEG quality factors. Our estimated bit-rates show a strong linear correlation coefficient of 0.98 with respect to the actual JPEG bit-rates. It is worth noting that as part of our overall training loss, a scalar weight is applied on the estimated bit-rates (μ in the total loss (5)). This means that the estimated bit-rats are only required to be accurate up to a scalar factor. Also, these results are computed for the low bit-rate range that we explore in this work (less than 1 bpp).

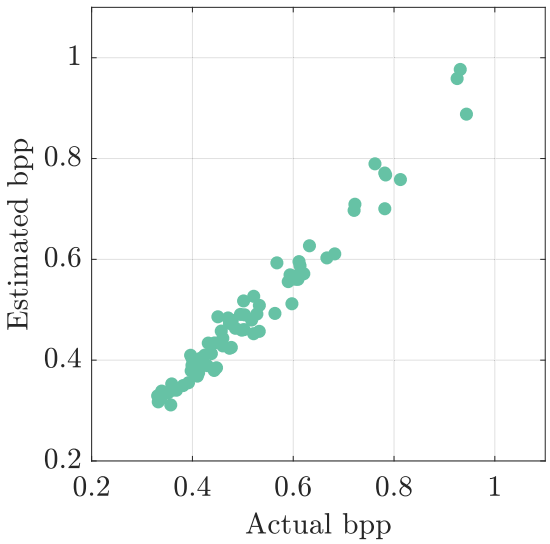


Fig. 3. Actual and estimated bpps for Kodak24 [22] images with various JPEG bit-rates. The linear correlation coefficient between the estimated and actual bpps is 0.98. Each point on this plot corresponds to one of Kodak24 images compressed with a specific JPEG quality factor (total of 72 data points, for 3 quality factors of 10, 15 and 20). Note that these estimated bpps correspond to $\mu B_q(.)$ in the total loss (5) with $\mu = 0.67$.

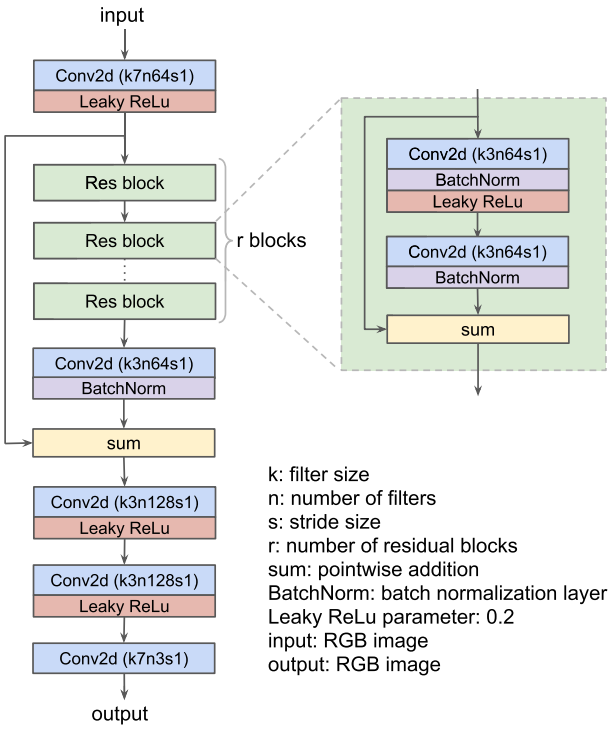


Fig. 4. Our image smoothing CNN.

5) Total Loss: To summarize, the following is the loss function we use in our experiments:

$$Loss(\Theta) = \sum_k \Big[dist\left(C_q(T(\Theta, \mathbf{z}_k)), \mathbf{z}_k\right) + \lambda Q\left(C_q(T(\Theta, \mathbf{z}_k))\right) + \mu B_q\left(C_q(T(\Theta, \mathbf{z}_k))\right) \Big]$$

where the distance function $dist(.)$ represents the perceptual error measures, the image quality $Q(.)$ is computed by NIMA

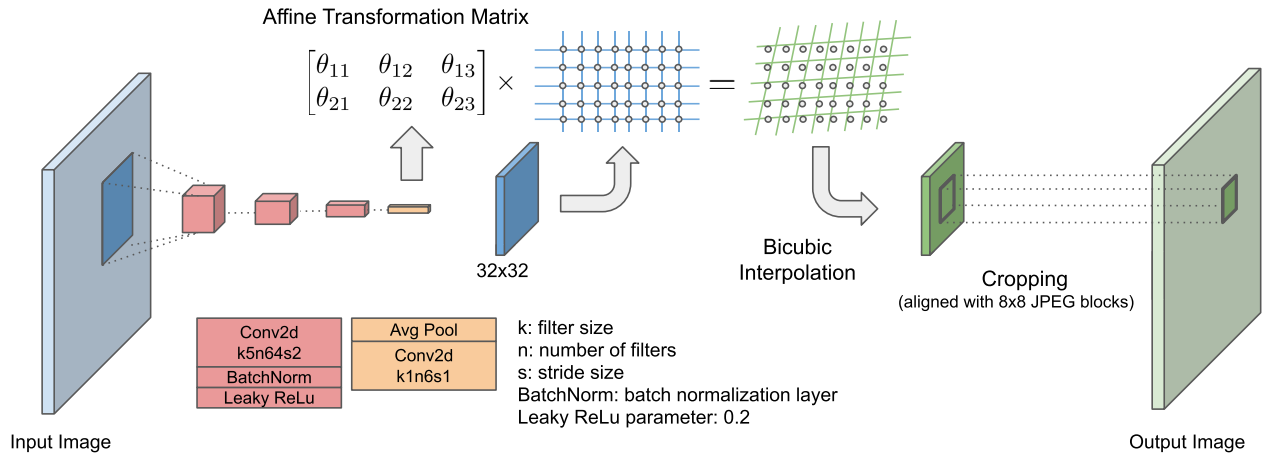


Fig. 5. Our patch-based spatial transformer network. The affine transformer parameters of 32×32 blocks are obtained from a trainable CNN. Transformed image grid is interpolated to obtain a warped image block of size 32×32. Finally, an 8×8 central block is extracted.

(a) Input (b) Smoothed input (c) Difference between (a) and (b)

Fig. 6. The difference between the input and the smoothed images (without JPEG compression). Our smoothing trained-network removes fine-grain details from the input image to make it more compressible by JPEG. Compressing (a) and (b) images with JPEG encoder at quality factor 20 takes 1.15 and 1.03 bpp, respectively.

and a total variation measure, and the entropy estimate $B_q(.)$ is computed over the quantized DCT coefficients of the edited image. Note that the same q-factor is applied both in the function $B_q(\cdot)$ and the differentiable JPEG function $C_q(\cdot)$.

B. The Editing Network

Our editing network consists of two parts; An image smoothing CNN (Fig. 4), and a patch-based warping operation (Fig. 5). While the smoothing is similar to a denoiser that controls fine-grained details, the spatial transformer allows for subtle local warps that make the underlying image more compressible [44]. More details on both parts is given below.

1) The Smoothing Network: Our image smoothing CNN is shown in Fig. 4. This convolutional neural network is similar to the residual CNN of Ledig *et al.* [38]. This architecture has $r = 2$ identical residual blocks, with 3×3 kernels and 64 feature maps followed by batch normalization [28] layers and Leaky ReLu (instead of parametric ones) [26] activations. To avoid boundary artifacts, the input image and feature maps are symmetrically padded before convolutions. We also append a smoothing strength factor (noise standard deviation) and the JPEG quality factor as extra channels to the input image. We observed that this additional information helps with better generalization of the model.

Examples of using the trained smoothing network are shown in Fig. 6. These images are not compressed by JPEG, and only represent edits applied to the input. The difference image shows that our editing removes fine details. Note that compressing the smoothed image with JPEG encoder at quality

(a) Input (b) Warped input (c) Difference of (a) and (b)

Fig. 7. The difference between the input and the warped images (without JPEG compression). Our warping makes spatial transformations on local image patches to make them more compressible by JPEG. Compressing (a) and (b) images with JPEG encoder at quality factor 20 takes 0.725 and 0.708 bpp, respectively.

factor 20 takes 1.03 bpp, whereas the same encoder takes 1.15 bpp for compressing the input image.

2) The Spatial Transformer Network (STN): As shown by Rott *et al.* [44], local image deformations can lead to DCT domain sparsity and consequently better compressibility. Unlike [44] that solves an alternating optimization with an optical flow, we use the differentiable spatial transformer network [29]. STN learns 6 parameters for an affine local transformation that allows cropping, translation, rotation, scale, and skew to be applied on the input (Fig. 5). We apply STN on overlapping blocks of size 32×32, and then we extract central crops of size 8×8 that are aligned with JPEG blocks. Since each 32×32 block is warped separately, this can cause inconsistency near the boundary of cropped blocks. To alleviate this, all overlapped grid values are averaged across neighboring blocks.

Examples of using the trained STN are shown in Fig. 7. The STN warps textures and edges locally to make the 8×8 blocks more compressible by JPEG encoder. Compressing the input and deformed images in Fig. 7(a) and Fig. 7(b) with JPEG encoder at quality factor 20 requires 0.725 bpp and 0.708 bpp, respectively.

To take advantage of both editing stages, we cascade the smoothing and warping operations. While the smoothing allows for less blockiness artifacts, the STN leads to texture preservation. Next, we discuss our training data.

C. Data

Our editing networks are trained on uncompressed images. To this end, we use burst processed images of Hasinoff *et al.* [27], which provides 3640 images of size 12 mega pixels. All images are converted to 8-bit PNG format. We extract about 120K non-overlapping patches of size 480×640, and use them to train our model. We also create a test set with 10% of the data.

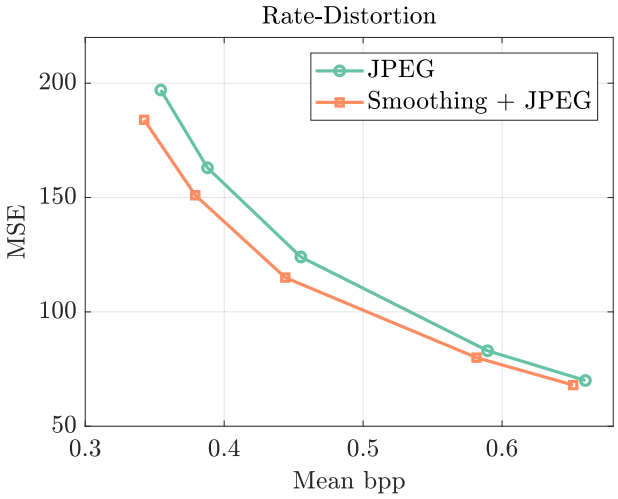


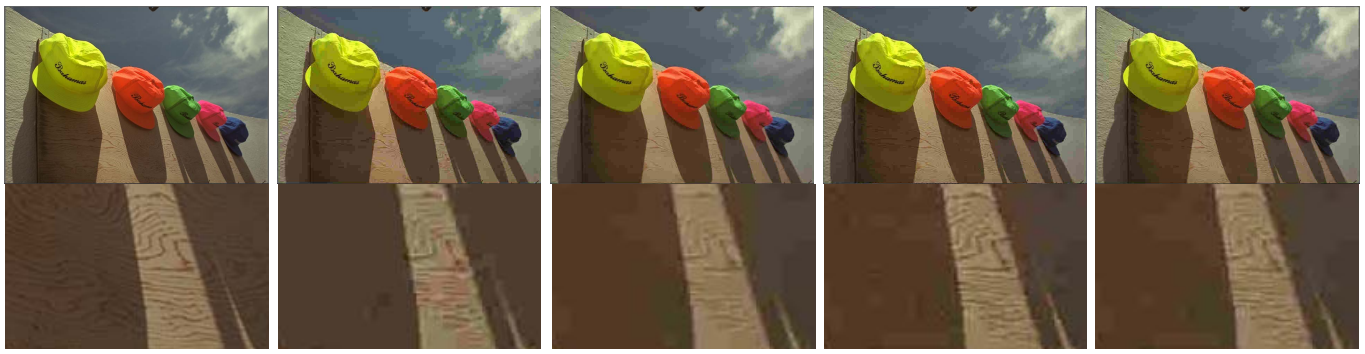
Fig. 8. MSE vs. mean bit-rate for the Kodak dataset [22].

IV. Relation to Prior Work

We pause our main story for a while and discuss the rich literature on combating compression artifacts. Our goal is to give better context to the suggested methodology by presenting the main existing alternatives. Note, however, that this does not constitutes an exhaustive scan of the existing literature, as this is beyond the scope of this work. We survey these algorithms by dividing them into categories based on their core strategies:

A. Post-Processing Algorithms

[4], [5], [14], [16], [20], [21], [33], [34], [37], [45], [51], [57], [60]–[62]: Those are the most common methods available, operating on the image after the compression-decompression damage has already been induced. Algorithms of this sort that are designed in the context of the JPEG format



(a) Input (b) Baseline JPEG (0.3529 bpp) (c) Smoothing+JPEG (0.3508 bpp) (d) STN+JPEG (0.3525 bpp) (e) STN+Smoothing+JPEG (0.3495 bpp)

Fig. 9. Compression performance with our proposed framework. (c) Smoothing the image before compression leads to less blockiness and color artifacts (0.6% bpp reduction), (d) The STN network generates more compressible details, (e) The combination of smoothing and STN lowers the bit-rate by 0.9% bpp. Our perceptual study shows that (e) is the most preferred image.

are known as deblocking algorithms. The idea behind these methods, be it for JPEG or any other transform-based coder, is quite simple, even though there are many ways to practice it; Given the compressed-decompressed image and knowing the quantization levels and the transform applied, the original image to be recovered must lie in a convex set that has a rotated hyper-rectangle shape. A recovery algorithm should seek for the most probable image within this set, something that could be done by relying on various regularization strategies. While some algorithms make use of this very rationale directly, others relax it in various ways, by simplifying the constraint set to a sphere, by forcing the recovery algorithm to take a specific shape, and more. At it simplest form, such a deblocking could be a simple linear filter applied to the boundaries between adjacent blocks.

B. Deep-Learning Based Solutions

[13], [19], [23], [56]: Still under the regime of post-processing, recent solutions rely on deep neural networks, trained in a supervised fashion to achieve their cleaning goal. These methods tend to be better performing, as their design targets the recovery error directly, instead of relying on model-based restoration methods.

C. Scale-Down and Scale-Up [12], [39], [52], [58]

An entirely different way to avoid compression artifacts is to scale-down the image before compression, apply the compression-decompression on the resulting smaller image, and scale-up the outcome in the client after decompression. This approach is especially helpful in low bit-rates, since the number of blocks is reduced, the bit stream overhead reduce along with it, and the scale-up at the client brings an extra smoothing. Variations over this core scheme have been proposed over the years, in which the scale-down or up are optimized for better end-to-end performance.

D. Pre-Processing Algorithms [36], [43], [48], [53]

It is well known that compression-decompression often behaves as a denoiser, removing small and faint details from the image. Nevertheless, applying a well-designed denoiser prior to the compression may improve the overall encoding performance by better prioritizing the content to be treated. The existing publications offering this strategy have typically relied on this intuition, without an ability to systematically design the pre-filter for best end-to-end performance, as the formulation of this problem is quite challenging.

E. Deformation Aware Compression [44]

While this work offers a pre-processing of the image along the same lines as described above, we consider it as a class of its own because of two reasons: (i) Rather than using a denoiser, the pre-process applied in this work is a geometrical warp, which re-positions elements in the image to better match the coder transform and block-division; and (ii) the design of the warp is obtained by an end-to-end approximate optimization method. Indeed, this paper has been the source of inspiration behind our ideas in this work.

Our proposed method joins the list of pre-processing based artifact removal algorithms, generalizing the work in [44] in various important ways: (i) our method could accommodate more general editing effects; (ii) its application is simple and fast, once the editing network has been trained; and (iii) we employ a no-reference image quality assessment that supports better quality outcomes. As already mentioned, the pre-processing strategy has a unique advantage over the alternative methods in the fact that the decoder does not have to be aware of the manipulations that the image has gone through, applying a plain decoding, while leaving the burden of the computations to the encoder. That being said, we should add that this approach can be easily augmented with a post-processing stage, for fine-tuning and improving the results further.

We conclude this survey of the relevant literature by referring to two recent and very impressive papers. The work reported in [10] offers a theoretical extension of the classic rate-distortion theory by incorporating the perceptual quality of the decompressed-image, exposing an unavoidable trade-off between distortion and visual quality. Our work practices this very rationale by sacrificing image content (via pre-editing) for

(a) Input (b) Baseline JPEG (1.0567 bpp) (c) Smoothing+JPEG (1.0423 bpp) (d) STN+JPEG (1.0485 bpp) (e) STN+Smoothing+JPEG (1.0445 bpp)

Fig. 10. Compression performance with our proposed framework. (c) Smoothing the image before compression leads to less blockiness and less details (1.3% bpp reduction), (d) The STN network generates more compressible details at lower bit-rate compared to the baseline (0.8% bpp reduction), (e) The combination of smoothing and STN lowers the bit-rate by 1.1% bpp. Our perceptual study shows that (d) is the most preferred image.

obtaining better looking compressed-decompressed images. The work by Agustsson et. al [3] offers a GAN-based learned compression algorithm that practically trades visual quality for distortion. While aiming for the same goal as our work, [3] replaces the whole compression-decompression process, whereas we insist on boosting available standard algorithms, such as JPEG, due to their massive availability and spread use.

V. Experimental Results

In this section our results are discussed and compared to other methods. Our train and test are performed on a single Nvidia GPU V100 with 16GB RAM. At training, images are cropped to 480×640, and testing is performed on the Kodak dataset [22]. We use the Adam optimizer [35] with learning rate set to 0.0001, and batch size as 1. The editing notworks are trained for 5×10^5 steps of stochastic gradient descent. Weights from the NIMA are kept fixed during training.

In order to train the STN and the smoothing network, we randomly sample the JPEG quality factor from a uniform distribution in the range $[8, 25]$ at each step of the gradient descent. This allows our editing to be effective for a range of bit-rates. At test time, we compare our results with the baseline JPEG at comparable bit-rates. To compress a test image at various bit-rates, we adjust the JPEG quality factor to ensure that our result compresses with the closest fewer bits.

We trained the smoother and STN networks separately, and then cascaded and fine-tuned them jointly. Next we discuss each model.

A. The Smoothing Network

The smoothing model is trained with an L_2 distance measure $dist(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$. The weights in our loss (Eq. 5) are set as $\lambda = 0$ and $\mu = 0.01$. Note that this weight selection allows for optimizing the typical rate-MSE curve. Training images are augmented with random additive white Gaussian noise to enforce smoothing property in the resulting network. We randomly vary the standard deviation of the noise in the range $[0, 0.15]$ at each training step, and append the noise standard deviation and JPEG quality factor as extra channels to the input RGB image. Rate-distortion curves of the regular JPEG and the smoother content are shown in Fig. 8. As expected, the smoothing improves upon the baseline JPEG. Note that these results are obtained before fine-tuning the smoother with the STN. Examples of the smoother editing are shown in Fig. 9, where color degradation and blockiness artifacts are more visible in the baseline JPEG, compared to our results.

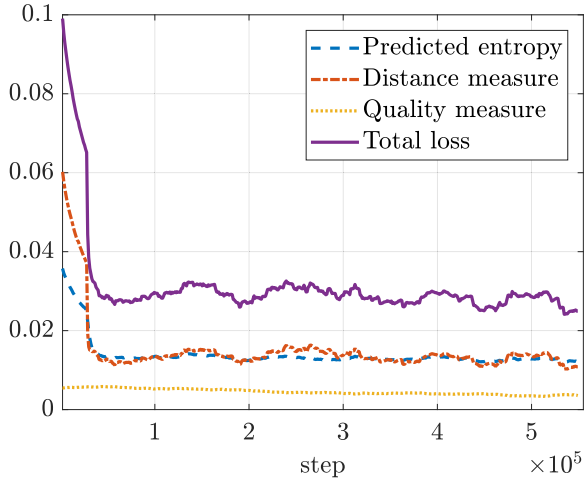
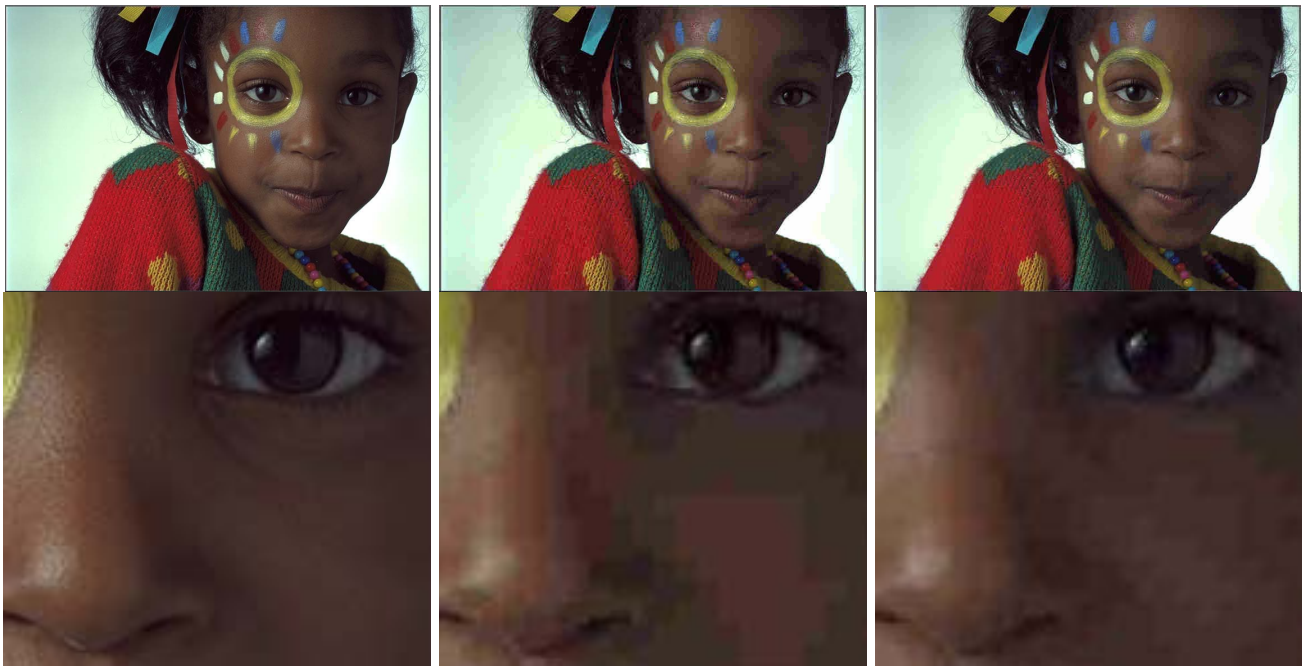


Fig. 11. Our training loss components during gradient descent with JPEG quality factor in the range [8,25]. For better display, all losses are smoothed.

B. The Spatial Transformer Network

The STN model is obtained by training with the distance measure $dist(\mathbf{x}_1, \mathbf{x}_2) = \|F(\mathbf{x}_1) - F(\mathbf{x}_2)\|_1$, where $F(.)$ represents the NIMA activations. This is due to the fact that L_1 or L_2 loss does not allow spatial transformations and warps. In contrast, NIMA CNN activations [50] are relatively tolerant to such transformations. The weights in our training loss (Eq. 5) are set as $\lambda = 0.02$ and $\mu = 1.0$. Our results for the STN network are shown in Fig. 10. Our editing of the input images allows to preserve structures and textures more effectively. The local deformations of the STN seem to make certain image textures more compressible. Note that this is a



(a) Input (b) Baseline JPEG (0.4705 bpp) (c) Smoothing + STN + JPEG (0.4355 bpp)

Fig. 12. Compression performance for applying smoothing and STN (7.4% bpp reduction).

different behavior than the smoother's effect. Also, it is worth noting that the spatial transformations can significantly reduce PSNR, and consequently rate-distortion analysis is not a fair performance evaluation of the STN.

C. The Cascaded Model

Our weighted losses for the cascaded smoother and STN are shown in Fig. 11. The weights in the training loss are similar to the weights used in the STN model. Our experiments suggest that the weighted predicted entropy and the distance measure should be close to each other. Also, as discussed in [49], the quality measure is most effective when its contribution is limited to a fraction of the total loss. We fine-tune both the smoother and STN networks jointly and present the results in Figs. 12 and 13. The cascade editor seems to present comparable details to the baseline, but with less JPEG artifacts.

We carried a human evaluation study to compare our proposed framework with baseline JPEG. We used Amazon Mechanical Turk with pairwise comparison for this task. We asked raters to select the image with better quality. We processed 24 images from Kodak dataset [22] with our smoothing and warping (STN) frameworks and compared them with their baseline JPEG counterparts at similar bit-rate. Comparisons are made by 20 human raters, and average percentage of the raters preference over baseline JPEG is reported in Fig. 14. As can be seen, both STN and our smoothing show perceptual preference of more than 50% for bit-rates smaller than 0.5 bpp. For higher bit-rates our methods did not provide a statistically significant advantage over baseline. Also, we observed that smoothing consistently outperforms STN.

We also compare our results with respect to the NIMA score [50] in Fig. 15. We computed NIMA scores for non-overlapping patches extracted from Kodak dataset and averaged the resulting scores. Fig. 15 indicates that the proposed pre-editing shows an advantage over the baseline JPEG for bpp $\in [0.45, 0.6]$. For bit-rates outside this range, the perceptual advantage over the baseline JPEG disappears.

D. Ablation Study

The impact of our design choices such as the perceptual loss and the model depth are discussed in the following. More explicitly, the CNN activations $F(.)$ in the perceptual loss $\|F(\mathbf{x}_1) - F(\mathbf{x}_2)\|_1$ are intermediate VGG-16 layers trained for quality assessment [50]. We tried the third, the fourth, and the fifth convolutional blocks as our feature functions to train various STN models. We observed that the average NIMA score obtained from the STN model trained with the fifth layer of VGG is 2.3% better than the model trained with the fourth layer activations. This gap increases to 4.0% when comparing STN models trained with the fifth and the third layer activations. These measurements are made for nearly comparable bit-rates.

Another important parameter in our framework is the depth of the smoothing CNN shown in Fig. 4. The depth parameter is effectively controlled by the number of residual blocks (r). We trained several models with various number of residual blocks. More specifically, we varied r from 1 to 5, and trained 5 different smoothing networks. Our experiments show that MSE of the models with $r = 3, 4, 5$ are very close, and the improvement from $r = 1$ to $r = 2$ is about 3.5%. Increasing r from 2 to 3 only showed a modest 0.2% improvement in the MSE.

(a) Input (b) Baseline JPEG (0.4293 bpp) (c) Smoothing + STN + JPEG (0.4169 bpp)

Fig. 13. Compression performance for applying smoothing and STN (2.9% bpp reduction).

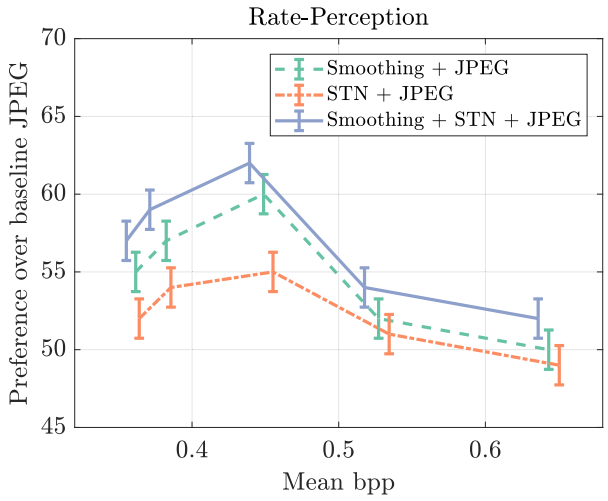


Fig. 14. Percentage of human raters preference for pairwise comparison between our result and baseline JPEG. Each data point is an average of 480 ratings (24 Kodak images [22] and 20 human raters).

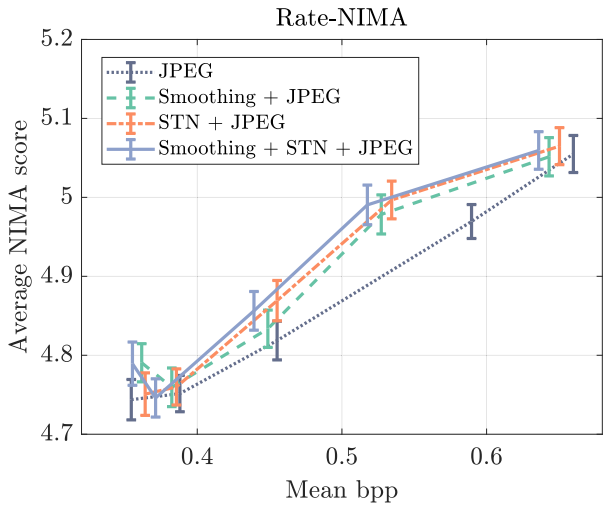


Fig. 15. Comparing our results with respect to the NIMA score [50]. These scores range from 1 to 10, with 10 indicating the highest quality. Each data point is an average of scores from non-overlapping crops extracted from Kodak images.

We conclude by referring to run time: We ran both our editors on an Intel Xeon CPU @ 3.5 GHz with 32 GB memory and 12 cores. We only measure timing of the pre-editing operation, as both methods use the same JPEG encoder. The smoothing CNN and STN run in 1.7 sec and 1.2 sec on a 1 mega pixel image, respectively. Since our editors are based

on convolutional neural networks, these running times can be further improved by GPU inference.

VI. Conclusion

One of the main bottlenecks of low bit-rate JPEG compression is loss of textures and details and presence of visual artifacts. In this work we have proposed an end-to-end trainable manipulation framework that edits images before compression in order to mitigate these problems. Our CNN-based trained editors optimize for better perceptual quality, lower JPEG distortions and color degradation. The proposed image-editors are trained offline, avoiding the need for per-image optimization, or a post-processing on the decoder (client) side. It is worth mentioning that JPEG is the dominating compression technology in digital cameras, cellphones, and webpages, with nearly 70% of all websites on the internet using images with JPEG format [2]. Thus, improvements in the JPEG standard may have significant industrial impact. Our future work will focus on extending this idea to other image compression standards, while seeking new ways to allow for more daring editing effects.

References

[1] (2013). *Requirements for Still Image Coding Using HEVC*. [Online]. Available: http://mpeg.chiariglione.org/standards/mpeg-h/high-efficiency-video-coding/requirements-still-image-coding-using-hevc

[2] (2020). *Usage Statistics of JPEG for Websites*. [Online]. Available: http://w3techs.com/technologies/details/im-jpeg

[3] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, "Generative adversarial networks for extreme learned image compression," 2018, *arXiv:1804.02958*. [Online]. Available: http://arxiv.org/abs/1804.02958

[4] F. Alter, S. Durand, and J. Froment, "Adapted total variation for artifact free decompression of JPEG images," *J. Math. Imag. Vis.*, vol. 23, no. 2, pp. 199–211, 2005.

[5] A. Z. Averbuch, A. Schclar, and D. L. Donoho, "Deblocking of block-transform compressed images using weighted sums of symmetrically aligned pixels," *IEEE Trans. Image Process.*, vol. 14, no. 2, pp. 200–212, Feb. 2005.

[6] J. Ballé, S. J. Hwang, N. Johnston, and D. Minnen. (2018). *Tensorflow Compression*. [Online]. Available: https://github.com/tensorflow/compression

[7] J. Ballé, V. Laparra, and P. E. Simoncelli, "End-to-end optimized image compression," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017. [Online]. Available: https://dblp.org/rec/conf/iclr/BalleLS17.html?view=bibtex

[8] J. Ballé, D. Minnen, S. Singh, S. Jin Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018, *arXiv:1802.01436*. [Online]. Available: http://arxiv.org/abs/1802.01436

[9] S. Beygi, S. Jalali, A. Maleki, and U. Mitra, "An efficient algorithm for compression-based compessed sensing," *Inf. Inference A, J. IMA*, vol. 8, no. 2, pp. 343–375, 2018.

[10] Y. Blau and T. Michaeli, "Rethinking lossy compression: The rate-distortion-perception tradeoff," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 675–685.

[11] S. Bosse, D. Maniry, T. Wiegand, and W. Samek, "A deep neural network for image quality assessment," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3773–3777.

[12] A. M. Bruckstein, M. Elad, and R. Kimmel, "Down-scaling for better transform compression," *IEEE Trans. Image Process.*, vol. 12, no. 9, pp. 1132–1144, Sep. 2003.

[13] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A deep convolutional neural network for image compression artifact suppression," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 752–759.

[14] T. Chen, H. R. Wu, and B. Qiu, "Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 5, pp. 594–602, May 2001.

[15] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.

[16] Y. Dar, A. M. Bruckstein, M. Elad, and R. Giryes, "Postprocessing of compressed images via sequential denoising," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3044–3058, Jul. 2016.

[17] Y. Dar, M. Elad, and A. M. Bruckstein, "Optimized pre-compensating compression," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 4798–4809, Oct. 2018.

[18] Y. Dar, M. Elad, and A. M. Bruckstein, "System-aware compression," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2226–2230.

[19] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 576–584.

[20] K. Du, H. Han, and G. Wang, "A new algorithm for removing compression artifacts of wavelet-based image," in *Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng.*, vol. 1. Jun. 2011, pp. 336–340.

[21] K. Du, J. Lu, H. Sekiya, and T. Yahagi, "Post-processing for restoring edges and removing artifacts of low bit rates wavelet-based image," *IEEJ Trans. Electron., Inf. Syst.*, vol. 127, no. 6, pp. 928–936, 2007.

[22] R. Franzen. (1999). *Kodak Lossless True Color Image Suite*. [Online]. Available: http://r0k.us/graphics/kodak

[23] L. Galteri, L. Seidenari, M. Bertini, and A. D. Bimbo, "Deep generative adversarial compression artifact removal," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4826–4835.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2414–2423.

[25] G. Ginesu, M. Pintus, and D. D. Giusto, "Objective assessment of the WebP image coding algorithm," *Signal Process., Image Commun.*, vol. 27, no. 8, pp. 867–874, 2012.

[26] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[27] S. W. Hasinoff *et al.*, "Burst photography for high dynamic range and low-light imaging on mobile cameras," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, Nov. 2016.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[29] M. Jaderberg *et al.*, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[30] B. Jin, M. V. O. Segovia, and S. Süsstrunk, "Image aesthetic predictors based on weighted CNNs," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 2291–2295.

[31] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.

[32] R. L. Joshi, M. Rabbani, and M. A. Lepley, "Comparison of multiple compression cycle performance for JPEG and JPEG 2000," in *Proc. 23rd Appl. Digit. Image Process.*, vol. 4115, Dec. 2000, pp. 492–501.

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

[34] T. Kartalov, Z. A. Ivanovski, L. Panovski, and L. J. Karam, "An adaptive POCS algorithm for compression artifacts removal," in *Proc. 9th Int. Symp. Signal Process. Appl.*, Feb. 2007, pp. 1–4.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] I Kopilovic and T Sziranyi, "Artifact reduction with diffusion preprocessing for image compression," *Opt. Eng.*, vol. 44, no. 2, Feb. 2005, Art. no. 027003.

[37] Y. Kwon, K. I. Kim, J. Tompkin, J. H. Kim, and C. Theobalt, "Efficient learning of image super-resolution and compression artifact removal with semi-local Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1792–1805, Sep. 2015.

[38] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.

[39] W. Lin and L. Dong, "Adaptive downsampling to improve image compression at low bit rates," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2513–2521, Sep. 2006.

[40] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang, "Deep multi-patch aggregation network for image style, aesthetics, and quality estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 990–998.
[41] L. Mai, H. Jin, and F. Liu, "Composition-preserving deep photo aesthetics assessment," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 497–506.
[42] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Trans. Image Process.*, vol. 21, no. 12, pp. 4695–4708, Dec. 2012.
[43] M. Oizumi, "Preprocessing method for DCT-based image-compression," *IEEE Trans. Consum. Electron.*, vol. 52, no. 3, pp. 1021–1026, Aug. 2006.
[44] T. R. Shaham and T. Michaeli, "Deformation aware image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2453–2462.
[45] M.-Y. Shen and C.-C. J. Kuo, "Review of postprocessing techniques for compression artifact removal," *J. Vis. Commun. Image Represent.*, vol. 9, no. 1, pp. 2–14, 1998.
[46] R. Shin and D. Song, "JPEG-resistant adversarial images," in *Proc. NIPS Workshop Mach. Learn. Comput. Secur.*, vol. 1, 2017.
[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[48] J.-L. Starck, F. Murtagh, B. Pirenne, and M. Albrecht, "Astronomical image compression based on noise suppression," *Publications Astronomical Soc. Pacific*, vol. 108, p. 446, May 1996.
[49] H. Talebi and P. Milanfar, "Learned perceptual image enhancement," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, May 2018, pp. 1–13.
[50] H. Talebi and P. Milanfar, "NIMA: Neural image assessment," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 399–4011, Apr. 2018.
[51] G. A. Triantaffilidis, D. Sampson, D. Tzovaras, and M. G. Strintzis, "Blockiness reduction in JPEG coded images," in *Proc. 14th Int. Conf. Digit. Signal Process.*, vol. 2, 2002, pp. 1325–1328.
[52] Y. Tsaig, M. Elad, P. Milanfar, and G. H. Golub, "Variable projection for near-optimal filtering in low bit-rate block coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 154–160, Jan. 2005.
[53] F. Tushabe and M. H. F. Wilkinson, "Image preprocessing for compression: Attribute filtering," in *Proc. Int. Conf. Signal Process. Imag. Eng. (ICSPIE)*, 2007, pp. 1411–1418.
[54] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2016, *arXiv:1607.08022*. [Online]. Available: http://arxiv.org/abs/1607.08022
[55] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.
[56] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang, "D^3: Deep dual-domain based fast restoration of JPEG-compressed images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2764–2772.
[57] P. Weiss, L. Blanc-Féraud, T. André, and M. Antonini, "Compression artifacts reduction using variational methods: Algorithms and experimental study," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2008, pp. 1173–1176.
[58] X. Wu, X. Zhang, and X. Wang, "Low bit-rate image compression via adaptive down-sampling and constrained least squares upconversion," *IEEE Trans. Image Process.*, vol. 18, no. 3, pp. 552–561, Mar. 2009.
[59] W. Xue, L. Zhang, and X. Mou, "Learning without human scores for blind image quality assessment," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 995–1002.
[60] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 1, pp. 91–95, Mar. 1992.
[61] G. Zhai, W. Lin, J. Cai, X. Yang, and W. Zhang, "Efficient quadtree based block-shift filtering for deblocking and deringing," *J. Vis. Commun. Image Represent.*, vol. 20, no. 8, pp. 595–607, Nov. 2009.
[62] 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. Image Process.*, vol. 22, no. 12, pp. 4613–4626, Dec. 2013.

Hossein Talebi received the B.S. and M.S. degrees in electrical engineering from the Isfahan University of Technology, Iran, and the Ph.D. degree in electrical engineering from the University of California at Santa Cruz, Santa Cruz, CA, USA. Since 2015, he has been with Google Research, Mountain View, CA, USA, where he works on computational imaging, image processing, and machine learning problems.

Damien Kelly received the B.A./B.A.I. degree in computer and electronic engineering from Trinity College Dublin in 2005 and the Ph.D. degree in 2010.

Since 2010, he has been working as a Research Fellow with Media Processing Group (www.sigmedia.tv), Trinity College Dublin, and Green Parrot Pictures Ltd., developing software tools for video enhancement. In 2011, he joined Google, where he has worked in chrome media and youtube video infrastructure teams on VR audio, audio/video transcoding, and quality enhancement. In 2018, he joined Computational Imaging Team, Google Research, where he is currently working on image/video super-resolution and enhancement.

Xiyang Luo received the B.S. degree in mathematics from Zhejiang University, Zhejiang, China, and the Ph.D. degree in applied mathematics from the University of California at Los Angeles, Los Angeles, CA, USA. Since 2019, he has been with Google Research, Mountain View, CA, USA, where he worked on image processing and machine learning.

Ignacio Garcia Dorado received the M.S. degree in electrical engineering from UPM, Spain, the M.S. degree in computer engineering from Lund University, Sweden, and the joint M.S. and Ph.D. degree in computer science from Purdue University, in 2008, 2014, and 2015, respectively.

From 2008 to 2010, he worked as a Computer Engineer at ESA, The Netherlands. From January 2010 to May 2010, he worked as a Research Assistant at McGill University, Canada. After this, he was awarded a Fulbright Scholarship to initiate Ph.D. studies. During his Ph.D. studies, he worked as a Research Intern at Nvidia during the Summer of 2013 and a Research Assistant at the University of California at Berkeley during the Summer of 2014. After his Ph.D. defense in October 2015, he moved to Mountain View to work with the Computational Photography Team, Google Research.

Feng Yang (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in automatic control from Tsinghua University, Beijing, China, in 2004 and 2007, respectively, and the Ph.D. degree in communication systems from the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2012.

He was a Research Assistant with Broadband Network and Digital Multimedia Laboratory, Tsinghua University, and Audiovisual Communications Laboratory, EPFL. He interned at Intel China Research Center, Beijing, and Nokia Research Center, Palo Alto, CA, USA. He was a Postdoctoral Researcher at Illumination and Imaging Laboratory, The Robotics Institute, Carnegie Mellon University (CMU), Pittsburgh, PA, USA. He is currently a Tech Lead Manager and a Senior Staff Software Engineer at Google Research, Mountain View, CA, USA. His research interests include deep learning, image and video quality assessment, processing, compression, security, and computational photography.

Dr. Yang was a recipient of the Fritz Kutter Award for his Ph.D. thesis "Bits from Photons: Oversampled Binary Image Acquisition" in 2012. He served as the Area Chair for CVPR 2020 and an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING and IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING.

Peyman Milanfar (Fellow, IEEE) received the bachelor's degree in electrical engineering and mathematics from the University of California, Berkeley, and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology.

From 1999 to 2014, he was a Professor of electrical engineering with UC Santa Cruz. From 2010 to 2012, he was an Associate Dean of research with the School of Engineering. From 2012 to 2014, he was on leave at Google-X, where he helped to develop the imaging pipeline for Google Glass. He is currently a Principal Scientist/Director at Google Research, where he leads the Computational Imaging Team. His team at Google developed the digital zoom pipeline for the pixel phones, which includes the multi-frame super-resolution (Super Res Zoom) pipeline (blog and project website), and the RAISR upscaling algorithm, and the night sight mode on pixel 3 uses our Super Res Zoom technology to merge images (whether you zoom or not) for vivid shots in low light, including astrophotography. He holds 15 patents, several of which are commercially licensed. He founded MotionDSP, which was acquired by Cubic Inc. (NYSE:CUB).

Dr. Milanfar along with his students, he has won several best paper awards from the IEEE Signal Processing Society. He has been a Keynote Speaker at numerous technical conferences, including Picture Coding Symposium (PCS), SIAM Imaging Sciences, SPIE, and the International Conference on Multimedia (ICME). He is a Distinguished Lecturer of the IEEE Signal Processing Society, and a fellow of the IEEE "for contributions to inverse problems and super-resolution in imaging."

Michael Elad (Fellow, IEEE) received the B.Sc., M.Sc., and D.Sc. degrees in electrical engineering from the Technion–Israel Institute of Technology, in 1986, 1988, and 1997, respectively.

After several years in industrial research, he served as a Research Associate with Stanford University from 2001 to 2003, working closely with Prof. Gene Golub (CS), Prof. Peyman Milanfar (EE-UCSC), and Prof. David L. Donoho (Stat). Since 2003, he holds a permanent faculty position with the Computer-Science Department, Technion–Israel Institute of Technology. He works in the field of signal and image processing, specializing in particular on inverse problems, sparse representations, and machine learning. He has authored 100 of technical publications in leading venues, many of which have led to exceptional impact. He is the author of the 2010's book *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, which is a leading publication in this field.

Prof. Elad has been a SIAM Fellow since 2018. He received numerous teaching and research awards and grants. He was awarded an ERC Advanced Grant in 2013. He was a recipient of the 2008 and 2015 Henri Taub Prizes for academic excellence, the 2010 Hershel-Rich prize for innovation, and three IEEE awards in 2018, such as the IEEE Signal Processing Society (SPS) Technical Achievement Award for contributions to sparsity-based signal processing, the IEEE SPS Sustained Impact Paper Award for his K-SVD paper mentioned above, and the SPS Best Paper Award for his paper on the Analysis K-SVD. He has served as an Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP) from 2007 to 2011, IEEE TRANSACTIONS ON INFORMATION THEORY (TIT) from 2011 to 2014, *Applied and Computational Harmonic Analysis* (ACHA) from 2012 to 2015, and *SIAM Journal on Imaging Sciences* (SIIMS) from 2010 to 2015. He held a Senior Editorial Role of IEEE SIGNAL PROCESSING LETTERS from 2012 to 2014. Since January 2016, he has been serving as the Editor-in-Chief for *SIAM Journal on Imaging Sciences* (SIIMS).