What's in the Image? Explorable Decoding of Compressed Images

Yuval Bahat and Tomer Michaeli
Technion - Israel Institute of Technology, Haifa, Israel
{yuval.bahat@campus,tomer.m@ee}.technion.ac.il

Abstract

The ever-growing amounts of visual contents captured on a daily basis necessitate the use of lossy compression methods in order to save storage space and transmission bandwidth. While extensive research efforts are devoted to improving compression techniques, every method inevitably discards information. Especially at low bit rates, this information often corresponds to semantically meaningful visual cues, so that decompression involves significant ambiguity. In spite of this fact, existing decompression algorithms typically produce only a single output, and do not allow the viewer to explore the set of images that map to the given compressed code. In this work we propose the first image decompression method to facilitate user-exploration of the diverse set of natural images that could have given rise to the compressed input code, thus granting users the ability to determine what could and what could not have been there in the original scene. Specifically, we develop a novel deep-network based decoder architecture for the ubiquitous JPEG standard, which allows traversing the set of decompressed images that are consistent with the compressed JPEG file. To allow for simple user interaction, we develop a graphical user interface comprising several intuitive exploration tools, including an automatic tool for examining specific solutions of interest. We exemplify our framework on graphical, medical and forensic use cases, demonstrating its wide range of potential applications.

1. Introduction

With surveillance systems so widely used and social networks ever more popular, the constant growth in the capacity of daily captured visual data necessitates using lossy compression algorithms (*e.g.* JPEG, H.264), that discard part of the recorded information in order to reduce storage space and transmission bandwidth. Over the years, extensive research has been devoted to improving compression techniques, whether by designing better decoders for existing encoders, or by devising new compression-decompression (CODEC) pairs, that enable higher perceptual quality even at low bit-rates. However, in any lossy compression method, the decoder faces inevitable ambiguity. This ambiguity is particularly severe at low bit-rates, which are becoming more prevalent with the ability to maintain perceptual quality at extreme compression ratios [1]. This is exemplified in Fig. 1 in the context of the JPEG standard. Low bit-rate compression may prevent the discrimination between different animals, or the correct identification of a shirt pattern, a barcode, or text. Yet, despite this inherent ambiguity, existing decoders do not allow the user to explore the abundance of plausible images that could have been the source of a given compressed code.

Recently, there has been growing research focus on models that can produce diverse outputs for any given input, for image synthesis [2, 3, 4], as well as for image restoration tasks, *e.g.* denoising [5], compression artifact reduction [6] and super-resolution [7, 8, 9]. The latter group of works took another step, and also allowed users to interactively traverse the space of high-resolution images that correspond to a given low-resolution input. In this paper, we propose the first method to allow users to explore the space of natural images that corresponds to a compressed image code. We specifically focus on the ubiquitous JPEG standard, though our approach can be readily extended to other image and video compression formats.

A key component of our method is a novel JPEG decompression network architecture, which predicts the quantization errors of the DCT coefficients and is thus guaranteed to produce outputs that are consistent with the compressed code. This property is crucial for enabling reliable exploration and examining what could and what could not have been present in the underlying scene. Our scheme has a control input signal that can be used to manipulate the output. This, together with adversarial training, allows our decoder to generate diverse photo-realistic outputs for any given compressed input code.

We couple our network with a graphical user interface (GUI), which lets the user interactively explore the space of

Figure 1. **Ambiguity in JPEG decompression.** A compressed JPEG file can correspond to numerous different plausibly looking images. These can vary in color, texture, and other structures that encode important semantic information. Since multiple images map to the same JPEG code, any decoder that outputs only a single reconstruction, fails to convey to the viewer the ambiguity regarding the encoded image.

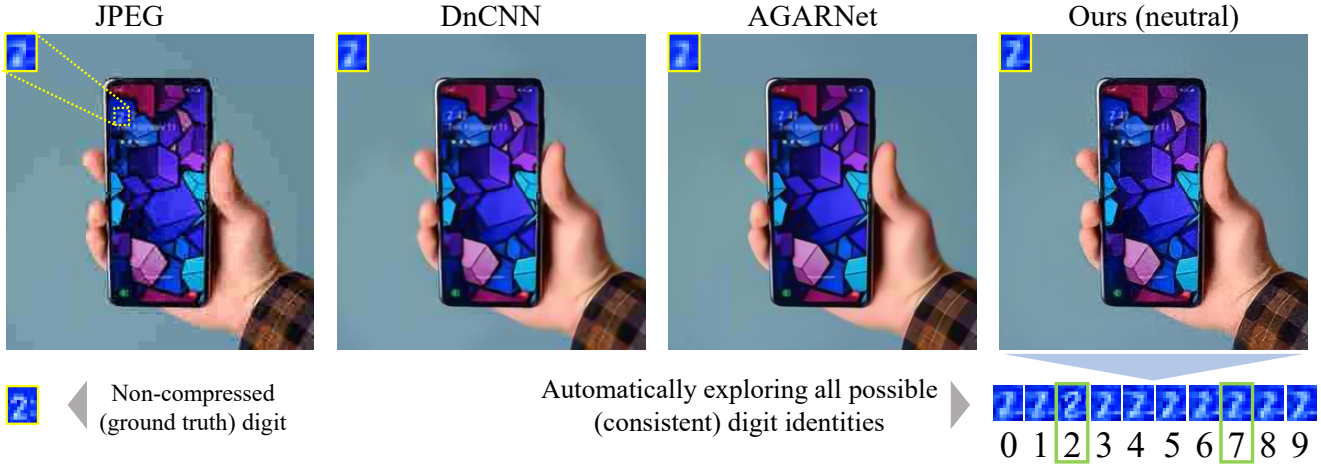

Figure 2. **Automatic exploration.** Upon marking an ambiguous character in the image, our GUI harnesses a pre-trained digit classifier to propose optional (consistent) reconstructions corresponding to the possible digits $0-9$ (see details in Sec. 5). This feature is valuable in many use cases (*e.g.* forensic), as it can assist in both revealing and resolving decompression ambiguities; although pre-exploration decoding of the hour digit (yellow rectangle) by all methods (top row) may suggest it is 7, our automatic exploration tool produces perceptually plausible decodings as both 7 and 2 (green rectangles), thus uncovering the hidden ambiguity and even preventing false identification, as the correct hour digit (in the pre-compressed image, bottom left) was indeed 2.

consistent and perceptually plausible reconstructions. The user can attempt to enforce contents in certain regions of the decompressed image using various tools (see *e.g.* Fig. 3). Those trigger an optimization problem that determines the control signal best satisfying the user's constraints. Particularly, our work is the first to facilitate *automatic* user exploration, by harnessing pre-trained designated classifiers, *e.g.* to assess which digits are likely to correspond to a compressed image of a digital clock (see Fig. 2).

Our explorable JPEG decoding approach is of wide applicability. Potential use cases range from allowing a user to restore lost information based on prior knowledge they may have about the captured image, through correcting unsatisfying decompression outputs (demonstrated in Fig. 8), to situations where a user wants to test specific hypotheses regarding the original image. The latter setting is particularly important in forensic image analysis and in medical image analysis, as exemplified in Figs. 2 and 4, respectively.

2. Related Work

Diverse and explorable image restoration Recently, there is growing interest in image restoration methods that can generate a diverse set of reconstructions for any given input. Prakash *et al.* [5] proposed to use a variational autoencoder for diverse denoising. Guo *et al.* [6] addressed diverse decompression, allowing users to choose between different decompressed outputs for any input compressed image. In the context of super-resolution, the GAN-based PULSE method [10] can produce diverse outputs by using different latent input initializations, while the methods in [7, 8, 9] were the first to allow user manipulation of their super-resolved outputs. Note that among these methods, only [7] guarantees the consistency of all its outputs with the low-resolution input, which is a crucial property for reliable exploration, *e.g.* when a user is interested in assessing the plausibility of a specific solution of interest.

Though we borrow some ideas and mechanisms from ex-

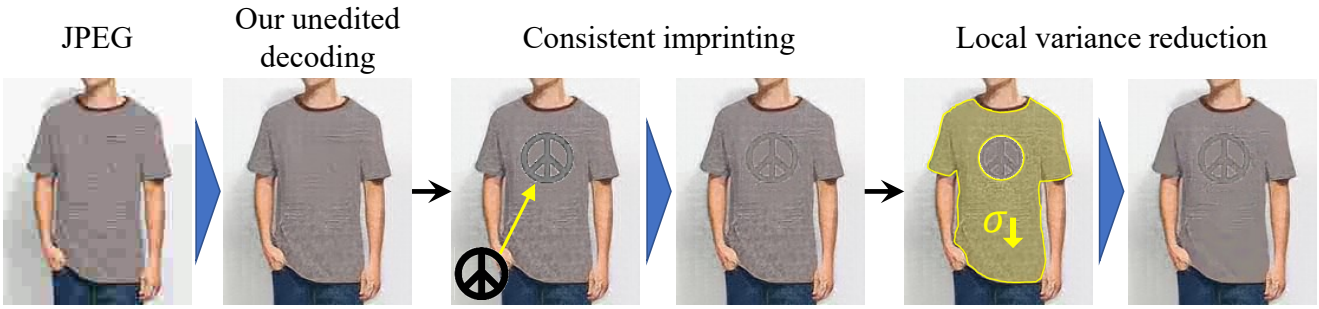

Figure 3. **Example exploration process.** Our GUI enables the user to explore the enforcement of various properties on any selected region within the image. Unlike existing editing methods that only impose photo-realism, ours seeks to conform to the user's edits while also restricting the output to be *perfectly consistent* with the compressed code.

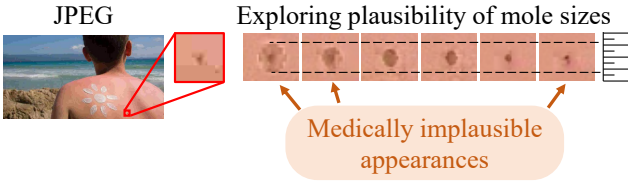

Figure 4. **Medical application example.** A dermatologist examining a suspected mole on a new patient may turn to existing patient photos containing this mole, to study its development. As the mole appearance in such images may often be degraded due to compression, our method can assist diagnosis by allowing exploration of the range of possible mole shapes and sizes. Please see corresponding editing processes in supplementary.

plorable super-resolution [7], this work is the first to discuss the need and propose a framework for performing *explorable image decompression*, which is a fundamentally more challenging task. While in super-resolution, the set of consistent solutions is a linear subspace that has zero volume in the space of all high-resolution images (just like a 2D plane has zero volume within 3D space), image compression involves quantization and so induces a set of consistent reconstructions with nonzero volume (just like a cube occupies nonzero volume in 3D space). We therefore introduce various novel mechanisms, including a fundamentally different consistency enforcing architecture, novel editing tools tailored for image decompression, and an *automatic* exploration tool (see Fig. 2), that is invaluable for many applications (e.g. forensics). Though ours is the first JPEG decompression method aiming for perceptual quality that is guaranteed to generate consistent reconstructions, we note that Sun *et al*. [11] proposed a consistent decompression scheme, but which is aimed at minimizing distortion rather than maximizing photo-realism (thus outputting the mean of the plausible explanations to the input).

Improved JPEG decompression Many works proposed improved decompression techniques for existing compression standards [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Specifically for JPEG, classical artifact reduction (AR) methods [12, 13, 14] attempted different heuristics, like smoothing DCT coefficients [13] or relying on natural image priors like sparsity, in both DCT and pixel domains [14]. Deep convolutional AR networks (first proposed by Dong *et al*. [22]) learn to minimize a reconstruction error with respect to ground truth reference images, and operate either in the pixel [22, 23, 19, 20], DCT [24, 11] or both domains [25, 26, 6, 27, 21]. Some recent AR methods [28, 29, 30] use a generative adversarial network (GAN) scheme [31] for encouraging more photo-realistic results, which we too employ in our framework. Our design (like [19, 20, 21]) is oblivious to the quality factor (QF) parameter, and can therefore handle a wide range of compression levels. In contrast, other methods are trained for a fixed QF setting, which is problematic not only because it requires training a different model for each QF, but also since QF by itself is an ambiguous parameter, as its conversion into compression level varies across implementations.

3. Our Consistent Decoding Model

To enable exploration of our decompression model's outputs, we need to verify they are both perceptually plausible and consistent with the given compressed code. To satisfy the first requirement, we adopt the common practice of utilizing an adversarial loss, which penalizes for deviations from the statistics of natural images. To satisfy the consistency requirement, we introduce a novel network design that is specifically tailored for the JPEG format. The JPEG encoding scheme works in the $Y-Cb-Cr$ color space and uses separate pipelines for the luminance (Y) and chrominance (Cb, Cr) channels. Our model supports color images, however for clarity we start by describing the simpler case of gray-scale images. The non-trivial treatment of color is deferred to Sec. 4. We begin with a brief description of the relevant components in the JPEG compression pipeline, before describing our network design.

3.1. JPEG compression

The encoding process is shown at the left hand side of Fig. 5. It starts by dividing the input image x, which is

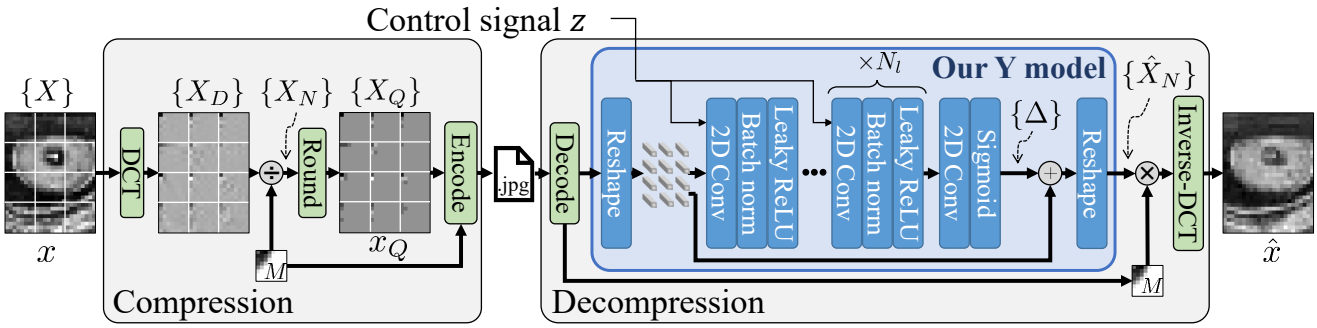

Figure 5. **Gray-scale JPEG compression scheme and our model.** Our network (inside the blue rectangle) is incorporated into the JPEG decompression pipeline in a way that guarantees the consistency of its outputs with the JPEG file content, while yielding a significant perceptual quality gain compared to images decompressed using the standard pipeline. An additional input signal z is incorporated to allow manipulating the output. Please refer to the description in Sec. 3.

assumed to be of size[1] $8m \times 8n$, into an $m \times n$ array of non-overlapping 8×8 blocks. For each 8×8 block X, the encoder computes its DCT, $X_D = \mathrm{DCT}(X)$, and divides it element-wise by a pre-defined matrix $M \in \mathbb{Z}^{8\times 8}$ to obtain a block of normalized DCT coefficients $X_N = X_D \oslash M$. Finally, the entries of X_N are rounded to yield a block of quantized coefficients $X_Q = \mathrm{round}(X_N)$. The blocks $\{X_Q\}$ are stored into the JPEG file alongside matrix M using some additional *lossless* processing steps. Note that the matrix M comprises the per-coefficient quantization intervals, determined as a function of the scalar QF parameter. However, this function varies between JPEG implementations, and therefore the QF itself is ambiguous and insufficient for extracting the image.

3.2. Our decoder design

Our decoder network is shown at the right hand side of Fig. 5. Our network operates in the DCT domain. Namely, for each 8×8 block X_Q extracted from the file, our network outputs an estimate $\hat{X}_D$ of the corresponding block X_D. The decoded image is then constructed by applying inverse DCT on the predicted DCT blocks $\{\hat{X}_D\}$. To predict X_D, we first generate a prediction $\hat{X}_N$ of the normalized coefficients X_N, and then multiply it element-wise by M, so that $\hat{X}_D = \hat{X}_N \odot M$. Since information loss during image encoding is only due to the rounding step, we consider a reconstructed block $\hat{X}_N$ to be consistent with the quantized block X_Q when it satisfies $\hat{X}_N = X_Q + \Delta$, with an 8×8 matrix Δ whose entries are all in $[-0.5, 0.5)$. We therefore design our network to predict this Δ for each block, and we confine its entries to the valid range using a shifted Sigmoid function. This process guarantees that the decoded image is perfectly consistent with the compressed input code.

Predicting the residual Δ for each block X_Q is done as follows. We arrange the blocks $\{X_Q\}$ to occupy the channel dimension of an $m \times n \times 64$ tensor x_Q, so that each block retains its relative spatial position w.r.t. the other blocks in the image. We then input this tensor to a network comprising N_ℓ layers of the form 2-D convolution $\rightarrow$ batch normalization $\rightarrow$ leaky ReLU, followed by an additional 2-D convolution and a Sigmoid. All convolution kernels are 3×3. The last convolution layer outputs 64 channels, which correspond to the residual blocks $\{\Delta\}$. Compared to operating in the pixel domain, the receptive field of this architecture is $8\times$ larger in each axis, thus allowing it to exploit larger scale cues.

An important distinction of our network is the ability to manipulate its output, which facilitates our goal of performing explorable image decompression. This is enabled by incorporating a control input signal, which we feed to the network in addition to the quantized input x_Q. We define our control signal $z \in \mathbb{R}^{m\times n\times 64}$ to have the same dimensions as x_Q, so as to allow intricate editing abilities. Following the practice in [7], we concatenate z to the input of each of the N_ℓ layers of our network, to promote faster training.

We train our model following the general procedure of [7]. As an initialization step, we train it to minimize the L_1 distance between ground truth uncompressed training images, and the corresponding outputs of our network, while randomly drawing the QF parameter for each image. Once initialized, training continues without utilizing any full-reference loss terms like L_1 or VGG, which is made possible thanks to the inherent consistency guarantee of our design. These full-reference loss terms are known to bias the output towards the (overly-smoothed) average of all possible explanations to the given compressed code, and are thus not optimal for the purpose of exploration. We instead minimize the following weighted combination of loss terms to guide our model:

$$\mathcal{L}_{\text{Adv}} + \lambda_{\text{Range}}\mathcal{L}_{\text{Range}} + \lambda_{\text{Map}}\mathcal{L}_{\text{Map}}. \quad (1)$$

Here, $\mathcal{L}_{\text{Adv}}$ is an adversarial loss, which encourages the re-

[1] We assume integer m and n only for simplicity. Arbitrary image sizes are also supported.

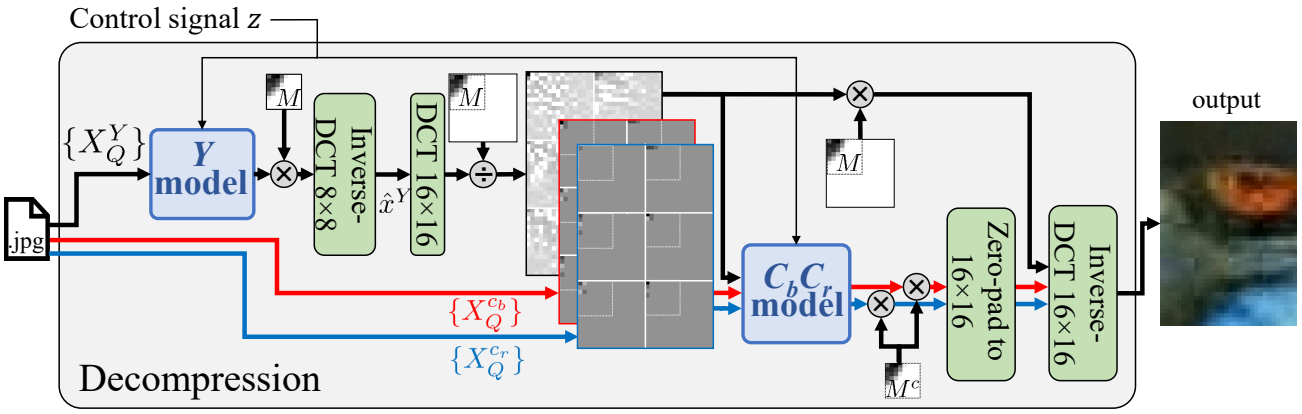

Figure 6. **Our full image decompression scheme.** We employ separate models (blue rectangles) to compensate for the quantization errors of the luminance and chroma channels. Both models share the same internal design depicted in Fig. 5, and receive the same control signal z, to allow coordinated editing. Please refer to the full description in Sec. 4.

constructed coefficient blocks $\hat{X}_D$ to follow the statistics of their natural image counterparts. In particular, we employ a Wasserstein GAN loss with spectral normalization [32] and gradient penalty [33], and use the same model architecture for both generator and critic (except for substituting batch normalization with layer normalization in the latter), following the recommendations in [33]. The second loss term, $\mathcal{L}_{\text{Range}}$, helps prevent model divergence by penalizing for pixel values outside $[16, 235]$, which is the valid range of luminance values in the $Y - Cb - Cr$ color space. We use $\mathcal{L}_{\text{Range}} = \frac{1}{k}\|\hat{x} - \text{clip}_{[16,235]}\{\hat{x}\}\|_1$, where $k = 64 \cdot m \cdot n$ is the number of pixels in the image.

The last loss term in (1) is associated with the control signal input z, which at exploration (test) time should allow traversing the space of plausible consistent decompressed images. We therefore use $\mathcal{L}_{\text{Map}} = \min_z \|\psi(x_Q, z) - x\|_1$ to (i) prevent our network ψ from ignoring its input z, as well as to (ii) guarantee our network can produce each ground truth natural image x in our training set using *some* z. This is in addition to the adversarial loss $\mathcal{L}_{\text{Adv}}$, which encourages the network's output corresponding to *any* input z to be perceptually plausible. Within each training step, we first compute $z^* = \arg\min_z\{\mathcal{L}_{\text{Map}}\}$ using 10 iterations, and then use the fixed z^* for the minimization of all loss terms in (1).

3.3. Training details

We train our model with the Adam optimizer on 1.15M images from the ImageNet training set [34], using batches of 16 images. The initialization and consecutive training phases last 6 and 12 epochs and employ learning rates of 10^{-4} and 10^{-5}, respectively. Batches in the latter phase are fed twice, once with a random z and once with the optimal z^* minimizing $\mathcal{L}_{\text{Map}}$ (see details in the Supp.). We set λ_{Range} and λ_{Map} to 200 and 0.1, respectively. To create training input codes, we compress the GT training images utilizing a quantization interval matrix $M = 50 \cdot Q_{\text{baseline}}/\text{QF}$, where Q_{baseline} is the example baseline table in the JPEG standard [35] and QF is independently sampled from a uniform distribution over $[5, 49]$ for each image[2]. We use $N_\ell = 10$ layers for both the generator and the critic models, using 320 output channels for all convolution operations but the last. We employ a conditional critic, by concatenating the generator's input x_Q to our critic's input, as we find it to accelerate training convergence.

4. Handling Color Channels

Let us denote the channels of a color image x by x^Y, x^{Cb}, and x^{Cr}. The chroma channels (Cb and Cr) of natural images tend to contain mostly low frequency content. The JPEG format exploits this fact, by allowing to subsample those channels in the pixel space. The subsampled channels are then divided into 8×8 blocks, whose DCT coefficients are quantized using matrix M^c, similarly to the luminance channel. These quantized blocks, denoted by $\{X_Q^{Cb}\}$ and $\{X_Q^{Cr}\}$, are stored in the color JPEG file alongside their luminance channel counterparts $\{X_Q^Y\}$. This process results in lost chroma information, which like its luminance counterpart, may correspond to semantically meaningful visual cues. Our framework allows exploring both the luminance and the chrominance of the image, as depicted in Fig. 6.

Here we use the most aggressive "4:2:0" subsampling configuration of JPEG, corresponding to subsampling the chroma channels by a factor of 2 in both axes. We reconstruct the chroma channels using an additional network, which handles the chroma information loss due to quantization. While we can use the same process employed in the luminance case to handle the chroma quantization, accounting for subsampling requires some modifications. Be-

[2]QFs in the range $[50, 100]$ induce lower data loss, and are thus less interesting for explorable decoding. The effect of control signal z in such high QFs is only minor, thanks to our framework's consistency guarantee.

Figure 7. **Exploring plausible explanations.** Artifacts in the given compressed image (left) are removed by our method (middle-left) prior to any exploration. We can then use our GUI to produce different explanations to the kitten's attention, by imprinting, *e.g.* a tiny fly or a worm onto the unedited image. These alternative reconstructions perfectly match the JPEG code when re-compressed. Please refer to the Supp. for a visualization of the control signals z corresponding to each output.

Figure 8. **Correcting unpleasing decompression.** Existing artifact removal methods like DnCNN [19] (middle-left), are often able to ameliorate compressed images (left), but do not allow editing their output. In contrast, outputs by our method (middle-right) can be edited by a user to yield superior results (right), which are guaranteed to match the input JPEG code if re-compressed.

fore elaborating on these modifications, we begin by briefly describing the relevant steps in the JPEG chroma pipeline.

4.1. Alternative modeling of chroma subsampling

Ideally, we would have liked to inform the chroma reconstruction network of the luminance information, by concatenating the luminance and chroma codes. However, this is impractical due to the spatial dimension mismatch resulting from the chroma subsampling. To overcome this hurdle, we remodel the above described subsampling pipeline using an approximated pipeline as follows.

In natural image chroma channels, almost all content is concentrated at the low frequencies (*e.g.*, in the BSD-100 dataset [36], an average of 99.99998% of the energy in each 16×16 chroma channel DCT block, is concentrated in the upper-left 8×8 sub-block). For such low-frequency signals, the above mentioned subsampling procedure incurs neglieible aliasing. Thus, the 8×8 DCT blocks of the subsampled channels can be assumed (for all practical purposes) to have been constructed by first computing the DCT of each 16×16 block of the original chroma channels, and then extracting from it only the 8×8 block of coefficients corresponding to the low-frequency content. The rest of the process is modeled without modification. As we show in the Supplementary, the differences between images processed using the actual and approximated pipelines are negligible (*e.g.* the PSNR between the two is 88.9dB over BSD-100).

4.2. Modifying our design to support subsampling

Given a compressed input code, we first reconstruct the luminance channel, as described in Sec. 3. The reconstructed luminance image $\hat{x}^Y$ is then fed into a chroma decoding network together with the quantized chroma blocks from the JPEG file, to obtain the final decoded color image.

Since the quantized 8×8 blocks of the chroma channels in fact correspond to 16×16 blocks of the image, our network operates on 16×16 blocks. Specifically, for the luminance channel $\hat{x}^Y$, we compute DCT coefficients for each 16×16 block and reshape them into a tensor with $16^2 = 256$ channels. The 8×8 chroma blocks stored in the file are zero-padded to be 16×16 in size (so that the high frequencies are all zeros) and then also reshaped into tensors with 256 channels (see Fig. 6). The luminance tensor is concatenated with the chrominance tensors to form a single tensor with $3 \times 256 = 768$ channels[3]. This tensor is then fed into our chroma network, which uses the same architecture described in Sec. 3, only with 160 channels in the internal layers. This network yields error estimate blocks Δ

[3]In practice, we discard the channels corresponding to the zero-padding, which are all zeros.

of size 8×8, which are added to the quantized blocks X_Q^{Cb} and X_Q^{Cr}, and multiplied by M^c. The resulting blocks are zero-padded to 16×16 (setting the high frequencies to zero) and converted back to pixel-space using inverse DCT. These reconstructed chroma channels are then combined with the luminance channel to yield the reconstructed color image. We feed the same control input signal z to both luminance and chroma models, to allow a coordinated editing effect.

5. Exploration Tools and Use Cases

Having trained both luminance and chroma models, we facilitate user exploration by employing a *graphical user interface* (GUI) comprising different editing tools. Our GUI runs on an NVIDIA GeForce 2080 GPU, and allows interactive exploration in real time. Specifically, once a compressed image code x_Q is loaded from a JPEG file, a user can manipulate the output of our decoding network, $\hat{x} = \psi(x_Q, z)$, by first marking a region to be edited and then choosing among different available tools. Those enable the user to attempt enforcing various properties on $\hat{x}$. Each editing tool triggers a process of solving $z^* = \arg\min_z f(\psi(x_Q, z))$ behind the scenes, for some objective function f, which is optimized using the Adam optimizer. The result is a modified output image $\psi(x_Q, z^*)$, which is guaranteed to be consistent with the compressed code x_Q (due to our network's architecture) and to have a natural appearance (due to the parameters of ϕ which have been shaped at train time to favor natural outputs). Examples for such images $\hat{x}$ are depicted in Fig. 7.

We introduce a very useful automatic exploration tool, which given an image region, presents optional reconstructions corresponding to each digit $d \in \{0, \ldots, 9\}$, by utilizing $f(\cdot) = \text{Classifier}_d(\cdot)$, the output of a pre-trained digit classifier [37] corresponding to digit d (see example use case in Fig. 2). This tool can easily be extended to domains other than digits, by using the appropriate classifiers.

Besides classifier-driven exploration, we also borrow objective functions from [7] and modify them for the JPEG decompression case, as well as add several JPEG-specific objectives to allow tuning local hue and saturation. The full set of available objective functions facilitates a wide range of operations, including manipulating local image variance (*e.g.* using $f(\cdot) = (\text{Var}(\cdot) - c)^2$ for some desired variance level c), performing piece-wise smoothing (*e.g.* using $f(\cdot) = \text{TV}(\cdot)$), propagating patches from source to target regions, modifying periodic patterns and more.

Another particularly useful group of tools allows embedding many forms of graphical user input, including various scribbling tools (similar to Microsoft-Paint), modifying local image brightness and even imprinting visual content from an external image. These tools act in two phases (corresponding to the middle pair of images in Fig. 3). They first enforce consistency of the desired input with the compressed image code, by projecting the scribbled (or imprinted) image onto the set of images that are consistent with the compressed code x_Q. Namely, each block of DCT coefficients $X_D^{\text{scribbled}}$ of the scribbled image is modified into

$$X_D^{\text{scribbled}} \leftarrow \left(\text{clip}_{[-\frac{1}{2},\frac{1}{2}]}\left(X_D^{\text{scribbled}} \oslash M - X_Q\right) + X_Q\right) \odot M. \tag{2}$$

This is the left of the middle pair in Fig. 3. In the second phase, an optimization process over z traverses the learned natural image manifold, searching for the output image that is closest to the consistent scribbled input. This is the right of the middle pair in Fig. 3. Variants of these tools provide many other features, including automatically searching for the most suitable embedding location, from a consistency standpoint. Please refer to the supplementary material for detailed descriptions of all tools provided by our GUI.

Our exploration framework is applicable in many domains and use cases, which we demonstrate through a series of representative examples[4]. Fig. 8 depicts a visually unsatisfying decoded JPEG image (left). Utilizing an artifact removal method yields some improvement, but significant improvement is achieved by allowing a user to edit the image, harnessing specific prior knowledge about the appearance of sand dunes. Another important application involves exploring the range of plausible explanations to the compressed image code, like the different appearances of the shirt in Fig. 1 or the focus of the kitten's attention in Fig. 7. Our framework can also be used to investigate which details could have comprised the original image. This is particularly important in medical and forensic settings. We demonstrate examples of exploring corrupted text in Fig. 2, and examining a mole in a medical use case in Fig. 4. More examples can be found in the supplementary.

6. Experiments

This work primarily focuses on introducing explorable image decompression, which we demonstrate on various use cases. Nevertheless, outputs of our framework are perceptually pleasing even prior to any user exploration or editing, as we show in Fig. 9 in comparison with DnCNN [19] and AGARNet [21], the only AR methods handling a range of compression levels (like ours) whose code is available online. Fig. 10 (left) further demonstrates the perceptual quality advantage of our method (Ours, GAN), by comparing NIQE [38] perceptual quality scores[5], evaluated on the

[4]Compressed images in our examples are produced by applying the JPEG compression pipeline to uncompressed images, though our method is designed to allow exploration of existing compressed codes.

[5]The no-reference NIQE measure is most suitable for our case, as it does not take into account the GT uncompressed images corresponding to the input JPEG code, which are as valid a decoding as any other output of our network, thanks to its inherent consistency. Nonetheless, the advantage of our method remains clear even when considering full-reference perceptual quality metrics, as we show in the Supp. using LPIPS [39].

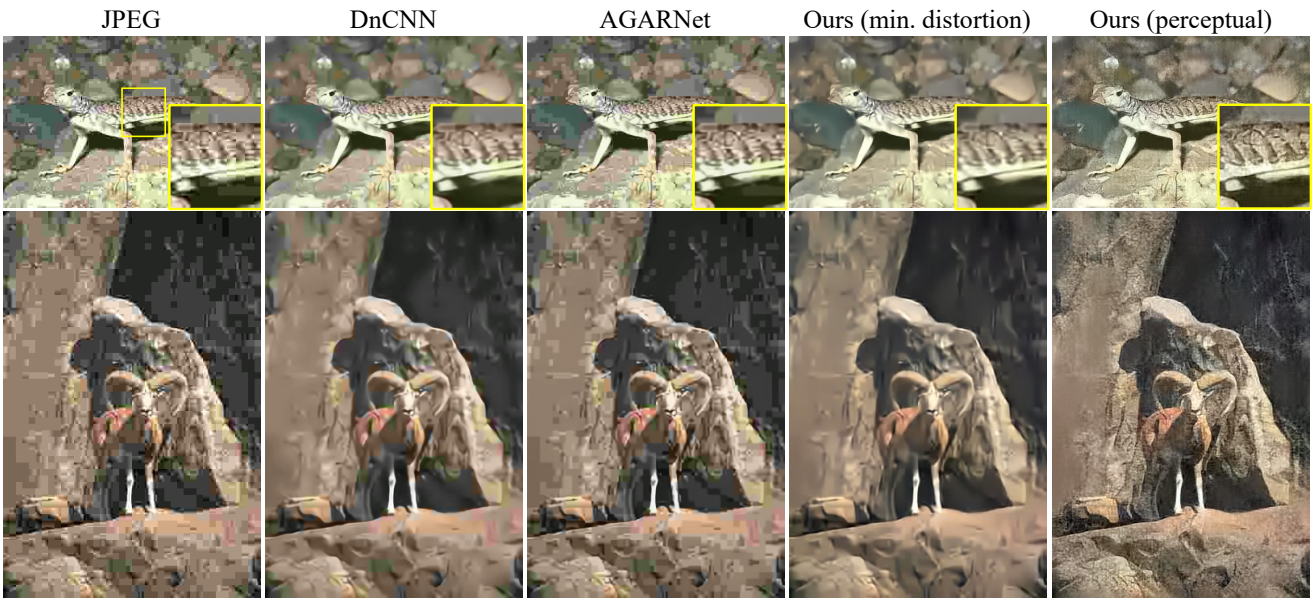

Figure 9. **Qualitative comparison.** Our GAN-trained image decompression model is primarily intended to allow consistent user exploration, especially for very low QFs, which induce larger ambiguities. Nonetheless, as demonstrated here on severely compressed images (QF=5), even pre-edited outputs (corresponding to random z inputs) of our model (right) yield significant perceptual quality gains relative to the JPEG decompression pipeline (left), as well as compared to results by the DnCNN [19] & AGARNet [21] AR methods, and a variant of our model trained to minimize distortion (middle columns). Please refer to the supplementary for additional visual comparisons.

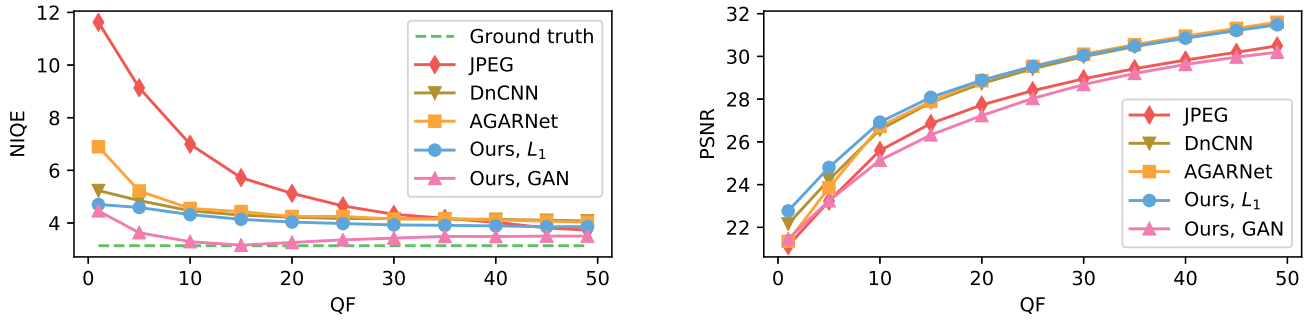

Figure 10. **Quantitative evaluation.** Comparing our exploration model (Ours, GAN), and its variant trained to minimize distortion (Ours, L_1), with the DnCNN [19] and AGARNet [21] AR methods on the BSD-100 [36] dataset, in terms of perceptual quality (left, lower is better) and image distortion (right, higher is better). Please refer to the Supp. for more comparisons and additional details.

BSD-100 [36] dataset. Finally, our consistency-preserving architecture produces state of the art results even when tackling the traditional (non-diverse, non-explorable) compression artifact removal task. Fig. 10 (right) presents a comparison between DnCNN and AGARNet, which aim for minimal distortion, and a variant of our model trained using only the L_1 penalty, *i.e.* the initialization phase in Sec. 3.2. Our model (Ours, L_1) compares favorably to the competition, especially on severely compressed images (low QFs). Please refer to the Supp. for additional details and comparisons, including evaluation on the LIVE1 [40] dataset.

7. Conclusion

We presented a method for user-interactive JPEG decoding, which allows exploring the set of naturally looking images that could have been the source of a compressed JPEG file. Our method makes use of a deep network architecture which operates in the DCT domain, and guarantees consistency with the compressed code by design. A control input signal enables traversing the set of natural images that are consistent with the compressed code. Like most decompression works, our framework is tailored to JPEG. However, the proposed concept is general, and can be applied to other compression standards, by identifying and addressing their respective sources of information loss. We demonstrated our approach in various use cases, showing its wide applicability in creativity, forensic, and medical settings.

Acknowledgements This research was supported by the Israel Science Foundation (grant 852/17) and by the Technion Ollendorff Minerva Center.

References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 221–231, 2019. 1

[2] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017. 1

[3] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018. 1

[4] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision*, pages 35–51, 2018. 1

[5] Mangal Prakash, Alexander Krull, and Florian Jug. Divnoising: Diversity denoising with fully convolutional variational autoencoders. *arXiv preprint arXiv:2006.06072*, 2020. 1, 2

[6] Jun Guo and Hongyang Chao. One-to-many network for visually pleasing compression artifacts reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3038–3047, 2017. 1, 2, 3

[7] Yuval Bahat and Tomer Michaeli. Explorable super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2716–2725, 2020. 1, 2, 3, 4, 7

[8] Marcel Christoph Bühler, Andrés Romero, and Radu Timofte. DeepSEE: Deep disentangled semantic explorative extreme super-resolution. *arXiv preprint arXiv:2004.04433*, 2020. 1, 2

[9] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. SRFlow: Learning the super-resolution space with normalizing flow. *arXiv preprint arXiv:2006.14200*, 2020. 1, 2

[10] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2445, 2020. 2

[11] Mengdi Sun, Xiaohai He, Shuhua Xiong, Chao Ren, and Xinglong Li. Reduction of jpeg compression artifacts based on dct coefficients prediction. *Neurocomputing*, 384:335–345, 2020. 3

[12] Kiryung Lee, Dong Sik Kim, and Taejeong Kim. Regression-based prediction for blocking artifact reduction in jpeg-compressed images. *IEEE Transactions on Image Processing*, 14(1):36–48, 2004. 3

[13] Tao Chen, Hong Ren Wu, and Bin Qiu. Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts. *IEEE transactions on circuits and systems for video technology*, 11(5):594–602, 2001. 3

[14] Xianming Liu, Xiaolin Wu, Jiantao Zhou, and Debin Zhao. Data-driven sparsity-based restoration of jpeg-compressed images in dual transform-pixel domain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5171–5178, 2015. 3

[15] Yehuda Dar, Alfred M Bruckstein, Michael Elad, and Raja Giryes. Postprocessing of compressed images via sequential denoising. *IEEE Transactions on Image Processing*, 25(7):3044–3058, 2016. 3

[16] Xianming Liu, Gene Cheung, Xiaolin Wu, and Debin Zhao. Random walk graph laplacian-based smoothness prior for soft decoding of jpeg images. *IEEE Transactions on Image Processing*, 26(2):509–524, 2016. 3

[17] Xueyang Fu, Zheng-Jun Zha, Feng Wu, Xinghao Ding, and John Paisley. Jpeg artifacts reduction via deep convolutional sparse coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2501–2510, 2019. 3

[18] Alexander G Ororbia, Ankur Mali, Jian Wu, Scott O'Connell, William Dreese, David Miller, and C Lee Giles. Learned neural iterative decoding for lossy image compression systems. In *Data Compression Conference*, pages 3–12. IEEE, 2019. 3

[19] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 3, 6, 7, 8

[20] Simone Zini, Simone Bianco, and Raimondo Schettini. Deep residual autoencoder for blind universal jpeg restoration. *IEEE Access*, 8:63283–63294, 2020. 3

[21] Yoonsik Kim, Jae Woong Soh, and Nam Ik Cho. Agarnet: Adaptively gated jpeg compression artifacts removal network for a wide range quality factor. *IEEE Access*, 8:20160–20170, 2020. 3, 7, 8

[22] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 576–584, 2015. 3

[23] Pavel Svoboda, Michal Hradis, David Barina, and Pavel Zemcik. Compression artifacts removal using convolutional neural networks. *arXiv preprint arXiv:1605.00366*, 2016. 3

[24] Jaeyoung Yoo, Sang-ho Lee, and Nojun Kwak. Image restoration by estimating frequency distribution of local patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2018. 3

[25] Zhangyang Wang, Ding Liu, Shiyu Chang, Qing Ling, Yingzhen Yang, and Thomas S Huang. D3: Deep dual-domain based fast restoration of jpeg-compressed images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2764–2772, 2016. 3

[26] Jun Guo and Hongyang Chao. Building dual-domain representations for compression artifacts reduction. In *Proceedings of the European Conference on Computer Vision*, pages 628–644. Springer, 2016. 3

[27] Xiaoshuai Zhang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. Dmcnn: Dual-domain multi-scale convolutional neural network for compression artifacts removal. In *IEEE International Conference on Image Processing*, pages 390–394. IEEE, 2018. 3

[28] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4826–4835, 2017. 3

[29] Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep universal generative adversarial compression artifact removal. *IEEE Transactions on Multimedia*, 21(8):2131–2145, 2019. 3

[30] Dong-Wook Kim, Jae-Ryun Chung, Jongho Kim, Dae Yeol Lee, Se Yoon Jeong, and Seung-Won Jung. Constrained adversarial loss for generative adversarial network-based faithful image restoration. *ETRI*, 41(4):415–425, 2019. 3

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 3

[32] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 5

[33] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. 5

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[35] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 5

[36] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 416–423, July 2001. 6, 8

[37] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013. 7

[38] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2012. 7

[39] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 7

[40] HR Sheikh. Live image quality assessment database release 2. *http://live. ece. utexas. edu/research/quality*, 2005. 8