Differentiable JPEG: The Devil is in the Details

Christoph Reich[1,2] Biplob Debnath[1] Deep Patel[1] Srimat Chakradhar[1]
[1]NEC Laboratories America, Inc. [2]Technische Universität Darmstadt
https://christophreich1996.github.io/differentiable_jpeg/

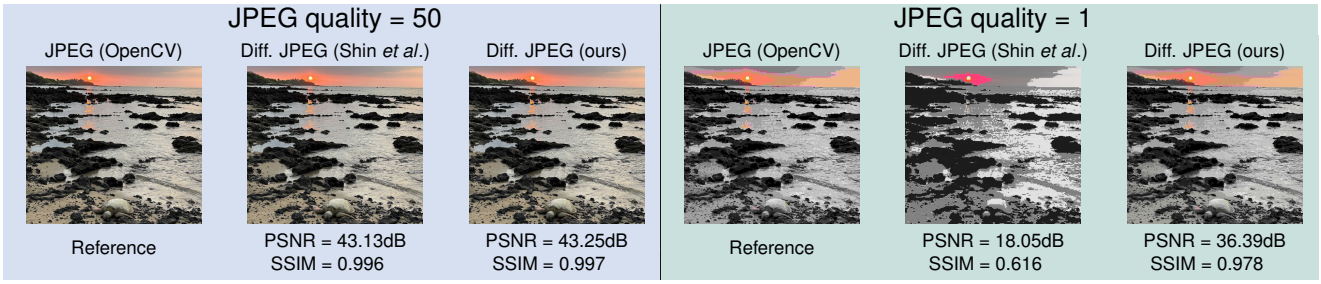

Figure 1. **Qualitative approximation results.** For a JPEG quality of 50, both Shin *et al*. [24] and our differentiable JPEG approach approximate the standard JPEG coding well. When reducing the JPEG quality to 1, the approach by Shin *et al*. does not approximate the JPEG coding well, while our differentiable JPEG still leads to a strong approximation. Structural similarity index measure (SSIM) and peak signal-to-noise ratio (PSNR) measured w.r.t. the coded image of the (non-differentiable) reference JPEG implementation (OpenCV [2]).

Abstract

JPEG remains one of the most widespread lossy image coding methods. However, the non-differentiable nature of JPEG restricts the application in deep learning pipelines. Several differentiable approximations of JPEG have recently been proposed to address this issue. This paper conducts a comprehensive review of existing diff. JPEG approaches and identifies critical details that have been missed by previous methods. To this end, we propose a novel diff. JPEG approach, overcoming previous limitations. Our approach is differentiable w.r.t. the input image, the JPEG quality, the quantization tables, and the color conversion parameters. We evaluate the forward and backward performance of our diff. JPEG approach against existing methods. Additionally, extensive ablations are performed to evaluate crucial design choices. Our proposed diff. JPEG resembles the (non-diff.) reference implementation best, significantly surpassing the recent-best diff. approach by 3.47dB *(PSNR) on average. For strong compression rates, we can even improve PSNR by* 9.51dB*. Strong adversarial attack results are yielded by our diff. JPEG, demonstrating the effective gradient approximation. Our code is available at* https://github.com/necla-ml/Diff-JPEG*.*

1. Introduction

JPEG (Joint Photographic Experts Group) coding is a standardized lossy compression approach for digital images [9, 30]. As one of the most popular image coding standards for storing and transmitting image data, JPEG coding has become an integral part of various devices and programs. The acceptable rate-distortion performance paired with a strong compression efficiency strikes a delicate balance for many applications. JPEG's straightforward implementation and support for parallel computing further bolsters its popularity and makes JPEG coding a preferred choice in many image processing pipelines [9].

The widespread use of JPEG in image processing pipelines has motivated the integration of JPEG coding into deep learning pipelines. Applications of JPEG coding in deep learning pipelines include (differentiable) data augmentation [7,12,13,23,25,26,37], data hiding [36,38], deepfake detection [34], adversarial attacks [6,24], or optimizing JPEG for deep neural networks [3, 16, 32]. Using JPEG in deep learning pipelines requires non-zero gradients to be propagated through the JPEG encoding-decoding. However, due to its inherently discrete nature, JPEG encoding-decoding is non-differentiable. Motivated by this, considerable effort has been devoted to building various differentiable JPEG approaches [3, 16, 24, 27, 32, 34, 36, 38].

While various differentiable JPEG approaches have been proposed, we are not aware of any work providing a comparison of these approaches. In this paper, we conduct a comprehensive review of existing differentiable JPEG approaches and highlight crucial deficiencies (*e.g*., not considering discretizations of standard JPEG) and suboptimal design choices (*e.g*., poor rounding approximations) that

impede precise predictions (*cf*. Fig. 1) and effective gradients. To remedy the outlined deficiencies, we present a novel differentiable JPEG approach. While drawing inspiration from prior differentiable JPEG implementations, our differentiable JPEG approach makes use of novel components, such as differentiable clipping, and is the first to model all important details of standard (non-differentiable) JPEG. In addition, we also propose a straight-through estimator (STE) variant of our differentiable JPEG.

We thoroughly evaluate the performance of existing differentiable JPEG implementations and our proposed approach, in approximating standard (non-diff.) JPEG coding. We show that all existing diff. approaches fail to accurately resemble standard JPEG over the whole compression range (*cf*. Fig. 1). To the best of our knowledge, our differentiable JPEG approach (w/ & w/o STE) is the first to provide an accurate approximation of standard JPEG across the entire range of compression strengths, while offering gradients w.r.t. all inputs. This is qualitatively showcased in Fig. 1. We validate the effectiveness of gradients derived from existing approaches and our differentiable JPEG by conducting adversarial attack experiments. Our approach generates superior adversarial samples in comparison to existing methods. These findings indicate that gradients obtained through our differentiable JPEG are notably more effective for gradient-based optimization tasks (*e.g.*, neural network training) than those derived from existing methods.

2. Background: The JPEG Coding Standard

The JPEG compression standard [10, 30], in the baseline mode, uses both lossy and lossless coding to achieve efficient image compression. The encoding starts by converting the original RGB image to the YCbCr color space and performing chroma downsampling. The YCbCr channels are then transformed into the frequency domain using a patch-wise discrete cosine transform (DCT). A given JPEG quality controls the quantization strength of the DCT features, trading file-size against distortion (*cf*. Sec. 2.3). Finally, the compressed JPEG file is produced using lossless coding. During decoding, the lossless and lossy encoding steps are reversed to reconstruct the JPEG-coded image from the JPEG file. Fig. 2 illustrates the JPEG coding process.

In general, JPEG encoding-decoding can be seen as a function mapping from an original (raw) RGB image $\mathbf{I}$ and the JPEG quality q to the JPEG-coded (distorted) image $\hat{\mathbf{I}}$

$$\begin{aligned} \mathrm{JPEG}\left(\mathbf{I}, \mathrm{q}\right) = \hat{\mathbf{I}}, \quad \mathrm{q} \in \{1, 2, \dots, 99\} \\ \mathbf{I}, \hat{\mathbf{I}} \in \{1, \dots, 255\}^{3 \times H \times W}. \end{aligned} \tag{1}$$

H and W denote the image resolution. Some implementations consider a max. q of 100, others of 99, for the sake of generality we use 99 as max. q. In the next subsections, we describe the internals of the JPEG function in detail.

2.1. JPEG encoding

The JPEG encoding process compresses a given image to a binary JPEG file and is composed of four main steps followed by lossless encoding (*cf*. Fig. 2). In the following, we explain the details of all encoding steps.

Color conversion (RGB $\rightarrow$ YCbCr). Digital imagery is typically displayed using the RGB color space. JPEG makes use of the YCbCr color space for compression. To this end, JPEG converts the RGB image to the YCbCr color space by a pixel-wise affine transformation.

Chroma subsampling. The human eye tends to be more sensitive to variations in brightness than to color details [17]. This motivates the use of chroma subsampling in JPEG. By discarding less relevant information to the human eye, chroma subsampling introduces a minimal loss in perceptual quality while leading to compression. Chroma subsampling is typically implemented by an anti-aliasing operation (*e.g.*, 2D convolution) followed by standard downsampling and is applied to both chroma channels (Cb & Cr).

Patch-wise discrete cosine transform. JPEG compression utilizes a patch-wise (and channel-wise) DCT-II operation to transform an image into a frequency (DCT) space. Before applying the DCT, non-overlapping 8×8 patches from the chroma-subsampled YCbCr image are extracted. For a given (flatten) patch $\mathbf{p} \in \{0, 1, \dots, 255\}^{64}$, the DCT is described by $\hat{\mathbf{p}} = \mathbf{a} \odot \mathbf{G}\mathbf{p}$. $\odot$ denotes the Hadamard product, $\mathbf{G} \in \mathbb{R}^{64 \times 64}$ contains the DCT coefficients, and $\mathbf{a}$ is a scaling factor. $\mathbf{G}$ is computed by $G_{8u+v,8i+j} = \cos\left(\frac{2x+1}{16}\right) \cos\left(\frac{2y+1}{16}\right)$ and $\mathbf{a}$ by $a_{8u+v} = \frac{1}{4}\alpha(u)\,\alpha(v)$ with $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$ and $u, v, i, j \in \{0, 1, \dots, 7\}$. $\hat{\mathbf{p}} \in \mathbb{R}^{64}$ represents the transformed patch. For simplicity, we omit the channel (YCbCr) and patch indexing.

Quantization. Through quantization, controlled by the JPEG quality q, frequencies are suppressed for the sake of compression. During the quantization step, the given JPEG quality q is mapped to a scale factor s by:

$$s(\mathrm{q}) = \begin{cases} \frac{5000}{\mathrm{q}} & \text{if } \mathrm{q} < 50 \\ 200 - 2\mathrm{q} & \text{otherwise.} \end{cases} \tag{2}$$

The scale factor is applied to the (standard) quantization table $\mathbf{QT}_{\mathrm{s}} \in \{1, 255\}^{8 \times 8}$ by $\hat{\mathbf{QT}} = \frac{s\,\mathbf{QT}_{\mathrm{s}} + 50}{100}$. The scaled quantization table is applied to each 2D DCT patch $\hat{\mathbf{P}} \in \mathbb{R}^{8 \times 8}$ (reshaped $\hat{\mathbf{p}}$) followed by the application of the rounding function $\overline{P}_{m,n} = \left\lfloor \frac{\hat{P}_{m,n}}{\hat{QT}_{m,n}} \right\rceil$ with $m, n \in \{0, 1, \dots, 7\}$. $\lfloor \cdot \rceil$ denotes the rounding to the next integer. Note that standard JPEG performs integer arithmetic to compute s and $\hat{\mathbf{QT}}$ this is equivalent to applying the floor function $\lfloor \cdot \rfloor$. Additionally, $\hat{\mathbf{QT}}$ is clipped to the integer range of

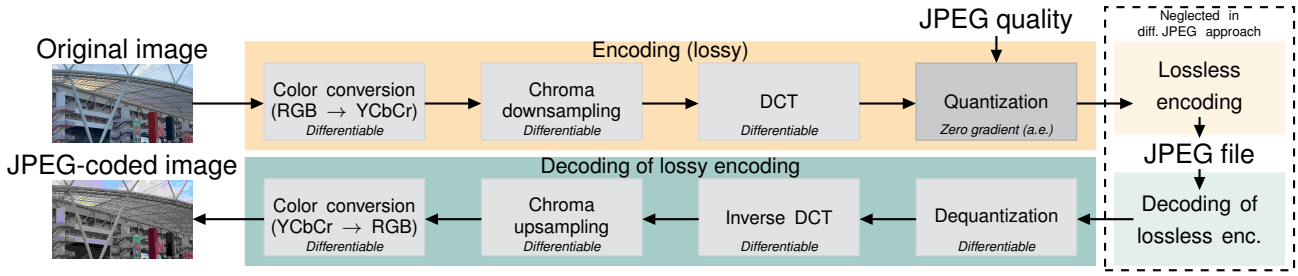

Figure 2. **The JPEG encoding-decoding pipeline.** The original input image is encoded to a JPEG file in a lossy manner. To recover the coded image the encoding is reversed in the decoding. JPEG uses lossless coding in conjunction with lossy coding. Since no information is lost (identity mapping) during the lossless encoding/decoding we can neglect these coding steps in our differentiable JPEG approach.

$\{1, 2, \ldots, 255\}$. Note JPEG also supports custom quantization tables and uses two separate tables for the luma channel (Y) as well as the chroma channels (Cb & Cr). For simplicity, we do not distinguish between quantization tables.

Lossless encoding. JPEG utilizes lossless entropy coding to compress all quantized DCT patches $\hat{\mathbf{P}}$. The lossless encoding first arranges the lossy encoded patches in a zigzag order before performing run-length encoding. Finally, Huffman coding is performed to build the binary JPEG file. Note the final JPEG file includes not only the encoded image content but also the scaled quantization tables and other markers including information such as the image resolution.

2.2. JPEG decoding

The JPEG decoding converts the compressed binary JPEG file back to an RGB image (*cf*. Fig. 2). Four main steps and the inversion of the lossless encoding operations compose the decoding. On a high level, every decoding step reverses the corresponding encoding step (*cf*. Fig. 2).

Decoding of lossless encoding. The Huffman-encoded JPEG file is decoded before the run-length encoding is undone. Finally, the information is rearranged as a pixel grid with three channels. Note that lossless encoding and decoding can be viewed as an identity mapping.

Dequantization. To dequantize, the quantized DCT features are multiplied with the respective scaled QT (luma or chroma table) $\tilde{\mathbf{P}} = \overline{\mathbf{P}} \odot \hat{\mathbf{Q}}\mathbf{T}$.

Inverse patch-wise discrete cosine transform. To convert the DCT information back into pixel space, the inverse discrete cosine transform is applied to each 8×8 patch.

Chroma upsampling. To recover the original image resolution, both chroma channels are upsampled using bilinear interpolation.

Color conversion (YCbCr $\rightarrow$ RGB). The coded image (YCbCr) is converted back into the RGB color space by applying the inverse of the previous affine transformation.

(a) Original image (b) JPEG quality 50 (c) JPEG quality 1

Figure 3. **JPEG coding artifacts.** *(a)* Original image, *(b)* JPEG-coded image with a JPEG quality of 50, file size is 47.3kB, and *(c)* coded image with a JPEG quality of 1, file size is 6.2kB. Image from the Set14 [35] and OpenCV [2] JPEG used.

2.3. JPEG rate-distortion trade-off

JPEG has strong support for different compression strengths. By adjusting the JPEG quality parameter q, different compression strengths can be achieved. A low JPEG quality results in a small file size but leads to significant image distortion (*cf*. Fig. 3c) since quantization suppresses plenty of frequencies. *Vice versa*, a high JPEG quality leads to a larger file size but reduces distortion (*cf*. Fig. 3b). This tradeoff is known as the rate-distortion trade-off.

JPEG employs the DCT in a patch-wise manner to achieve efficient compression. This compression approach introduces distinctive artifacts in the resulting distorted JPEG-coded image. These artifacts manifest in various forms, including ringing, contouring, posterizing, and most notably, block boundary artifacts. Among these artifacts, block boundary artifacts are particularly noticeable when compressing natural images (*cf*. Fig. 3b).

2.4. Non-differentiability of JPEG

The JPEG encoding-decoding process (*cf*. Eq. (1)) is an inherently discrete operation that precludes the application of continuous differentiation. However, we can extend all operations of the JPEG encoding-decoding to real-valued numbers. Consequently, we can formulate a continuous JPEG function, JPEG_c, that accepts continuous inputs in terms of the original image and JPEG quality and produces a continuous coded image ($\text{JPEG}_c : \mathbb{R}^{3\times H\times W} \times [1, 99] \rightarrow \mathbb{R}^{3\times H\times W}$). This naive

continuous generalization of the JPEG encoding-decoding suffers from a major limitation. The gradient of JPEG_c (w.r.t. to both inputs) is zero almost everywhere (a.e.) and undefined at points of jump discontinuity. This is caused by the reliance on rounding and floor functions in the encoding process. This property inhibits the direct integration of JPEG_c into gradient-based learning systems (*e.g.*, deep neural network training) as they rely on the availability of "useful" non-zero gradients for optimization.

3. Existing Differentiable JPEG Approaches

Existing differentiable JPEG approaches can be broadly categorized into three groups: straight-through estimator approaches, surrogate model approaches, and noise-based methods. All approaches aim to propagate "useful" gradients through the JPEG encoding-decoding.

STE approaches. In standard STE, the gradient of a non-differentiable function is approximated by assuming a constant gradient of one [1]. During the forward pass, the true non-differentiable function is used. STE has been shown to be effective in various deep learning-based approaches [1, 4, 11, 19, 29]. This technique is also used when the gradient of a function would be zero a.e. (*e.g.*, rounding function). Choi *et al*. [3] used STE to propagate gradients through the rounding operation of the JPEG encoding [28]. Instead of assuming a constant gradient as in standard STE, Xie *et al*. [32] utilizes the gradient of a tanh-based differentiable rounding approximation in the backward pass. Both approaches do not model the JPEG quality scaling of the quantization tables, limiting the general usability beyond the application proposed in the respective papers. Additionally, the bounded nature of the quantization tables and the quantization table scale is not considered. Nor is the bounded nature of the coded image modeled.

Surrogate model approaches. Another technique to achieve "useful" gradients is to replace all non-differentiable components of JPEG coding (*e.g.*, rounding) with differentiable approximations. The resulting differentiable JPEG approach is both an approximation in the forward and backward pass. Shin *et al*. proposed the first differentiable surrogate of the JPEG encoding-decoding [24]. A polynomial approximation of the rounding function $\lfloor x \rceil + (x - \lfloor x \rceil)^3$ enables the propagation of gradient through the rounding operation. Additionally, the quantization table scaling by the JPEG quality q is reformulated to $s(\mathrm{q}) = \begin{cases} \frac{50}{\mathrm{q}} & \text{if } \mathrm{q} < 50 \\ 2 - \frac{2\mathrm{q}}{100} & \text{otherwise} \end{cases}$ and $\hat{\mathbf{QT}} = s\mathbf{QT}_{\mathrm{s}}$. Note this leads to a difference of 0.5 in the scale quantization table $\hat{\mathbf{QT}}$ compared to the standard scaling factor computation (*cf*. Eq. (2)), subsequently deteriorating the approximation performance. Other approaches have been built on the approach by Shin *et al*. with slight modifications [16, 27, 33]. Instead of a polynomial approximation, Xing *et al*. approximate the rounding function with finite Fourier series approximation $x - \sum_{k=1}^{10} \frac{(-1)^{k+1}}{k\pi} \sin(2\pi k x)$. While Shin *et al*. does not model integer divisions nor the bounded nature of the quantization tables as well as the coded image, Xing *et al*. hard clips the coded image to the valid pixel range, leading to a zero-valued gradient for clipped pixels.

Noise-based approaches. JPEG coding introduces unique distortion artifacts to the coded image (*cf*. Sec. 2.3 and Fig. 3). This motivates noise-based approach, wherein JPEG coding is approximated by introducing specific noise into the original image. Zhu *et al*. [38] achieves this by randomly applying dropout to the DCT features mimicking JPEG distortion. However, applying random dropout offers very limited control over the precise quality, leads to sparse gradients, and is only a coarse and stochastic approximation. Zhang *et al*. [36] adds the true distortion as pseudo noise to the original image. While the resulting coded images match the true coded image, this approach is equivalent to applying STE to the full JPEG coding function. The resulting gradients are fully independent of the JPEG function. Additionally, both approaches only offer gradients w.r.t. the original image, severely limiting general applicability. Due to these major limitations, we do not consider noise-based approach as differentiable approximations.

JPEG file size modeling. Certain applications require a differentiable estimate of the JPEG file size [16, 32]. As all differentiable JPEG approaches neglect the lossless encoding/decoding, modeling the file size is non-trivial [16, 24, 32, 33]. Existing approaches typically train a deep neural network to regress the file size from the quantized DCT features of the differentiable JPEG approach [16, 32]. Note that the scope of this paper is to model JPEG encoding-decoding in a differentiable manner and not to model the JPEG file size. We refer the reader to Luo *et al*. [16] and Xie *et al*. [32] for more details on file-size modeling.

4. Method: Differentiable JPEG Coding

We aim to build a continuous and differentiable approximation ($\text{JPEG}_{\text{diff}} : [0, 255]^{3\times H\times W} \times [1, 99] \rightarrow [0, 255]^{3\times H\times W}$) of the full JPEG encoding-decoding function (*cf*. Eq. (1)). This approximation should accurately resemble standard (non-diff.) JPEG and yield gradients "useful" for gradient-based optimization. To achieve this, we first take a surrogate model approach (Sec. 4.1). Later we present the incorporation of the STE technique (Sec. 4.2).

We noticed that existing approaches just focus on finding "useful" differentiable surrogates of the rounding function used for quantization; however, other discretizations and bounds are not modeled. These operations include the clipping as well as the discretization of the quantization table, the discretization of the quantization table scaling, and the

bounding of the output/coded image. We present differentiable approximations for modeling all five operations. Note for operations not described (*e.g.*, DCT), we use their naive continuous generalization (*cf*. Sec. 2). Since the gradient is not affected by the lossless encoding/decoding (*cf*. Fig. 2), we follow existing approaches and neglect these steps.

4.1. Differentiable JPEG surrogate

Differentiable quantization. For approximating the quantization operation, which uses the rounding function, we utilize the polynomial approximation $\lfloor x \rceil + (x - \lfloor x \rceil)^3$ by Shin *et al*. [24]. While other approximations exist (*e.g.*, sigmoid func.), we later show that this design choice leads to superior performance over other approximations (*cf*. Sec. 6.3).

Differentiable QT scale floor. Non-differentiable standard JPEG computes the quantization table scaling $s(\mathrm{q})$ based on the JPEG quality with integer arithmetic (*cf*. Eq. (2)). This is equivalent to computing s with float precision and applying the floor function. To model this operation in a differentiable manner, we introduce a differentiable floor approach. Note the original scaling approach (*cf*. Eq. (2)) is used, not the approach by Shin *et al*. [24].

Our differentiable floor function makes use of the relation between the rounding and floor function. We can express the floor function as a shifted version of the rounding function $\lfloor x - 0.5 \rceil = \lfloor x \rfloor$. Based on this property, we can use the polynomial rounding approach to approximate the floor function by $\lfloor x - 0.5 \rceil + (x - 0.5 - \lfloor x - 0.5 \rceil)^3$. We later validate this design choice against other approximations (*cf*. Sec. 6.3).

Differentiable QT floor. Since the quantization table is included in every JPEG file, the JPEG standard requires the QT to include integer values. The standard (non-diff.) JPEG implementation ensures this by using integer arithmetic. This is equivalent to applying the floor function to QT after scaling. To ensure gradient propagation, we apply the proposed floor approximation to the scale QT.

Differentiable QT clipping. Based on the JPEG standard [30], the quantization table is bounded to the integer range $\{1, \dots, 255\}^{8\times 8}$. Utilizing low JPEG qualities (strong compression) can lead to values outside of this range, even when utilizing the standard QT. To ensure values approximately within this range, we propose a differentiable (soft) clipping operation $\overline{\mathrm{clip}}$.

$$\overline{\mathrm{clip}}(x) = \begin{cases} x & \text{if } x \in [b_{\min}, b_{\max}] \\ \gamma\, x & \text{otherwise} \end{cases}, \; x \in \mathbb{R}, \gamma \in (0, 1]. \quad (3)$$

This soft approximation ensures a non-zero gradient of x when outside of the range $[b_{\min}, b_{\max}]$. We set the scale parameter γ to 10^{-3}.

Differentiable output clipping. Similar to the input image, the output image is bounded to the pixel range of $\{0, \dots, 255\}$. Depending on the image content and the applied JPEG quality, values outside of this range can occur. To approximately adhere to this range, we also apply the proposed differentiable clipping to the output/coded image.

4.2. Differentiable JPEG coding with STE

Instead of differentiably approximating all discretizations and bounds both in the forward and backward pass, we can also take advantage of the STE technique. In our STE-based differentiable JPEG approach, we utilize the true rounding, floor, and clipping functions in the forward pass. However, instead of using a constant gradient of one, as done by standard STE, we employ the gradient of the proposed approximations during backpropagation. This approach leads to a reduced error of the forward function since the true function and not an approximation is used. We later show that our STE approach can be beneficial in certain settings. We also show that using the gradient of the proposed approximations is more effective than standard STE.

5. Evaluation

Evaluating a differentiable JPEG implementation comes in two different flavors. First, evaluating the performance of the forward mapping and second, the validation of gradients obtained from the differentiable JPEG approach. While evaluating the forward mapping is well-defined, validating the effectiveness of the backward function is non-trivial.

Forward function evaluation. We evaluate the performance of the forward mapping by measuring the similarity between the coded image obtained by the differentiable approach and the coded image of the reference implementation. We use the SSIM [31] and the PSNR for evaluation.

Backward function evaluation. We aim to showcase the "usefulness" of gradients obtained by the backward function. Taking inspiration from Shin *et al*. [24] we utilize adversarial attack experiments to showcase the quality and "usefulness" of gradients in the context of gradient-based optimization. Adversarial examples are crafted through the mapping $C\left(\mathrm{JPEG}_{\mathrm{diff}}\left(\mathbf{I}, q\right)\right) = \mathbf{p}$, composed of an ImageNet classifier C (*e.g.*, ResNet-50 [8]) and a differentiable JPEG function $\mathrm{JPEG}_{\mathrm{diff}}$, conditioned on a given JPEG quality q. We aim to craft an adversarial image $\mathbf{I}_{\mathrm{adv}}$, from the original image $\mathbf{I}$, s.t. the prediction $p \in [0, 1]^c$ over c classes is deteriorated. Note, while crafting the adversarial example using a differentiable JPEG approach we validate the effectiveness of the adversarial example by using the non-differentiable JPEG reference implementation.

We consider two adversarial attack techniques, the fast gradient sign method (FGSM) [5] and the iterative fast gradient sign method (IGSM) [14, 15]. FGSM in the non-targeted setting crafts an adversarial example by $\mathbf{I}_{\mathrm{adv}} = \mathbf{I} + \epsilon \cdot \mathrm{sign}(\Delta_{\mathbf{I}}[\mathcal{L}(y, C\left(\mathrm{JPEG}_{\mathrm{diff}}\left(\mathbf{I}, q\right)\right))])$, where $\mathcal{L}$ is the

cross-entropy loss and y the true class label. IFGSM performs FGSM for N iterations and uses $\frac{\epsilon}{N}$ as the update size. Both FGSM and IFGSM ensure that $\|\mathbf{I}_{\text{adv}} - \mathbf{I}\|_\infty \leq \epsilon$.

We argue that in order to generate an effective adversarial example, $\text{JPEG}_{\text{diff}}$ needs to produce "useful" gradients. The more effective the adversarial images are in deteriorating the prediction (measured by accuracy), the more "useful" the gradients of $\text{JPEG}_{\text{diff}}$ become. Note, since C consumes the JPEG-coded image the resulting gradient is also partly dependent on the forward performance of $\text{JPEG}_{\text{diff}}$.

Vanishing gradient evaluation. While it is not possible to directly measure the "usefulness" of gradients, we can evaluate the gradients' ability to adhere to desired properties in gradient-based optimization. For local and global minima, it is desirable that gradients vanish. While the exact position of local and global minima w.r.t. to a differentiable JPEG approach is not known, we can measure the gradient magnitude at positions that are desired to be local or global minima. Optimally, the gradient at these positions vanishes. In particular, we compute the L1 loss between the differentiable and the reference JPEG $\mathcal{L}_1(\text{JPEG}_{\text{diff}}(\mathbf{I}, \text{q}), \text{JPEG}_{\text{r}}(\mathbf{I}, \text{q}))$. Subsequently, estimate the gradient norms w.r.t. to both the JPEG quality $\|\Delta_{\text{q}}\mathcal{L}_1\|$ and the (internal) standard QT $\|\Delta_{\mathbf{QT}_{Y,C}}\mathcal{L}_1\|$ (luma and chroma table). We average the gradient norms over both the different integer JPEG quality ranges and the dataset $\mathcal{D}$.

6. Experiments

Datasets. For our adversarial attack experiments, we utilize 5k randomly chosen images of the ImageNet-1k validation set (ILSVRC 2012) [22]. For all other experiments, we use the Set14 dataset [35], composed of 14 RGB images ranging from natural to document-like images.

Implementation details. For all experiments, we utilize the OpenCV [2] JPEG implementation as a reference. We utilize Kornia [21] for computing the SSIM. The utilized SSIM patch size is 11. Adversarial attack experiments are conducted using a ResNet-50 from torchvision [18] supervised trained on ImageNet-1k. For IFGSM experiments, we utilize $N = 10$ iterations. The step parameter ϵ is varied between experiments. After each attack iteration, we hard clip the image to the valid pixel range of $[0, 255]$.

Baselines. We compare against the surrogate-based approaches by Xing *et al*. [33] and Shin *et al*. [24]. We also compare against the STE-based approach of Xie *et al*. [32]. Since Xie *et al*. are not modeling the JPEG quality, we extend the approach with the JPEG quality mapping of Xing *et al*. [33]. Both the approach of Shin *et al*. and Xie *et al*. offer no code, we have reimplemented both in PyTorch [20]. For the approach of Xing *et al*. [33], we use the official code. We noticed a bug in the official code (wrong **QT** transposition). We run experiments with the debug code. Note due to the very limited control (JPEG quality can not be set) and stochasticity, we do not consider noise-based approaches.

6.1. Forward function results

We evaluate the ability to resemble the reference JPEG implementation of existing approaches against our differentiable JPEG approach (w/ STE). Our approach outperforms existing approaches over the whole JPEG quality range (*cf*. Fig. 4). The performance gap between differentiable methods becomes smaller for high JPEG quality values, but still, our approach leads to superior performance.

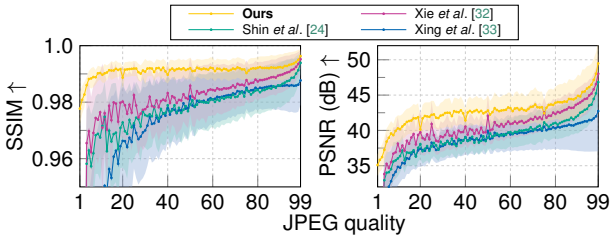

Figure 4. **Forward function performance.** Performance of approximating the reference JPEG implementation (OpenCV [2]) for different JPEG qualities. Mean & one standard deviation shown.

For small JPEG qualities, we observe a significant disparity in performance between methods. While our approach still approximates standard JPEG well, existing approaches fail and produce coded images vastly different from the reference implementation. This is showcased in Fig. 1 and quantitatively analyzed in Fig. 5 and Tab. 1.

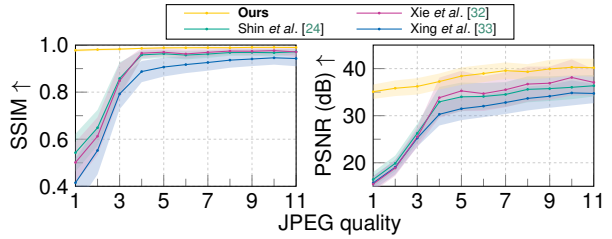

Figure 5. **Forward function performance for strong compression.** Performance of approximating the reference JPEG implementation (OpenCV [2]) for low JPEG qualities. Mean & one standard deviation shown.

Forward function ablation results. To understand what makes our differentiable JPEG implementation effective in resembling the reference implementation, we conduct an ablation study (*cf*. Tab. 1 and Fig. 6). We gradually add our introduced components (config. *A* to *F*). All our novel components improve performance while our full configuration (config. *E*) performs best among non-STE approaches. Using STE (config. *F*) further improves forward performance, leading to coded images perceptually indistinguishable from the reference implementation.

Fig. 6 showcases the effect of all introduced components for each (integer) JPEG quality. We observe that especially

Configuration	q range →	SSIM ↑ 1-99	SSIM ↑ 1-10	SSIM ↑ 11-99	PSNR ↑ 1-99	PSNR ↑ 1-10	PSNR ↑ 10-99
	Xing *et al*. [33]	0.961	0.833	0.977	38.10	29.45	39.19
	Xie *et al*. [32]	0.972	0.884	0.983	40.02	31.63	41.07
A	Shin *et al*. [24]	0.969	0.888	0.979	38.71	31.07	39.66
B	+ diff. QT clipping	0.978	0.966	0.979	39.16	35.10	39.67
C	+ diff. QT floor	0.983	0.971	0.985	41.03	35.95	41.66
D	+ diff. QT scale floor	0.984	0.971	0.986	41.08	35.96	41.72
E	+ diff. output clipping **(our differentiable JPEG)**	*0.991*	*0.987*	*0.992*	*42.60*	*38.28*	*43.14*
F	+ STE (*cf*. Sec. 4.2) **(our differentiable STE JPEG)**	**0.993**	**0.993**	**0.992**	**43.49**	**41.14**	**43.78**

Table 1. **Forward function performance summary & ablation.** To ablate our approach, we gradually add our novel components to Shin *et al*. [24]. We also report the performance of other diff. approaches. STE-based approaches marked in gray.

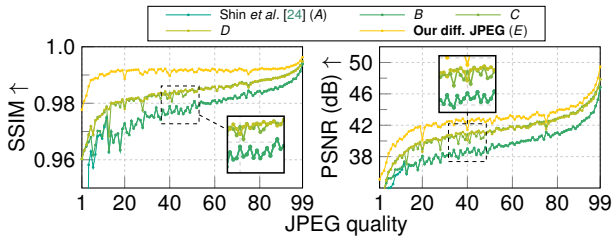

Figure 6. **Ablation study graph.** JPEG quality-wise forward function performance for different configurations (*cf*. Tab. 1).

differentially clipping the coded output image (config. *E*) improves performance and, in particular, leads to strong results for small JPEG qualities.

Using STE (config. *F*) leads to a superior forward function performance over our non-STE approach (config. *E*, *cf*. Fig. 6). As showcased in Fig. 7, our diff. STE JPEG approach especially outperforms our differentiable JPEG approach w/o STE for low JPEG qualities.

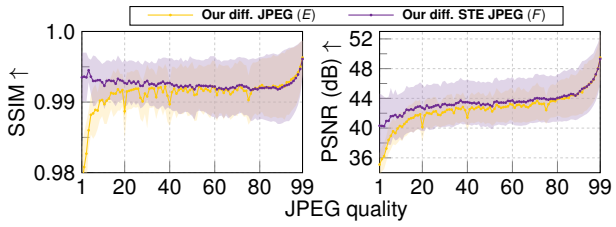

Figure 7. **Forward function performance with STE.** Performance of approximating the reference JPEG implementation (OpenCV [2]) for different JPEG qualties. Mean & one SD shown.

6.2. Backward function results

Adversarial attack results. We craft adversarial examples to showcase the "usefulness" of gradients derived by differentiable JPEG approaches. When using FGSM with $\epsilon = 3$, our differentiable JPEG w/o STE consistently leads to superior results over our approach w/ STE and other differentiable approaches (*cf*. Tab. 2 (top)). When increasing ϵ to 9, our differentiable JPEG w/ STE scores slightly better than our approach w/o STE (*cf*. Tab. 2 (bottom)).

Approach	q range →	Top-1 acc ↓ 1-99	Top-1 acc ↓ 1-10	Top-1 acc ↓ 11-99	Top-5 acc ↓ 1-99	Top-5 acc ↓ 1-10	Top-5 acc ↓ 10-99
No attack		66.83	33.36	71.01	85.91	53.10	90.01
FSGM with $\epsilon = 3$							
Xing *et al*. [33]		53.92	26.00	57.42	77.90	45.65	81.93
Xie *et al*. [32]		41.48	20.07	44.15	66.53	38.52	70.03
Shin *et al*. [24]		*36.62*	16.40	*39.15*	*60.19*	*32.38*	*63.67*
Our diff. JPEG		**36.51**	**15.80**	**39.10**	**59.92**	**30.89**	**63.54**
Our diff. STE JPEG		36.97	*16.38*	39.55	60.79	32.61	64.32
FSGM with $\epsilon = 9$							
Xing *et al*. [33]		51.25	27.64	54.20	75.24	47.41	78.72
Xie *et al*. [32]		38.29	18.15	40.81	61.89	35.38	65.21
Shin *et al*. [24]		35.01	15.04	*37.51*	57.40	*29.70*	60.86
Our diff. JPEG		*35.00*	**14.65**	37.55	*57.28*	**28.97**	*60.82*
Our diff. STE JPEG		**34.79**	*14.96*	**37.26**	**57.21**	29.86	**60.63**

Table 2. **FGSM attack results summary.** Summarized top-1 and top-5 accuracy after FGSM attack for different JPEG quality ranges. We report results for both $\epsilon = 3$ and $\epsilon = 9$. As a reference, we also report accuracies for no attack performed.

When using IFGSM to craft adversarial examples, the results are consistently in favor of our differentiable JPEG approach w/o STE (*cf*. Tab. 3). Interestingly, the approach by Xing *et al*. leads to predominately poor adversarial examples. We explain this result by the use of the Fourier rounding approximation which is highly non-monotonic.

Approach	q range →	Top-1 acc ↓ 1-99	Top-1 acc ↓ 1-10	Top-1 acc ↓ 11-99	Top-5 acc ↓ 1-99	Top-5 acc ↓ 1-10	Top-5 acc ↓ 10-99
IFSGM with $\epsilon = 3$							
Xing *et al*. [33]		43.44	24.42	45.82	72.52	45.55	75.90
Xie *et al*. [32]		25.30	14.72	26.63	46.55	31.47	48.43
Shin *et al*. [24]		15.11	8.98	15.88	27.21	19.99	*28.11*
Our diff. JPEG		**14.39**	**7.97**	**15.19**	**25.79**	**17.53**	**26.83**
Our diff. STE JPEG		*15.02*	*8.35*	*15.85*	*27.11*	*18.77*	28.15
IFSGM with $\epsilon = 9$							
Xing *et al*. [33]		39.59	24.99	41.41	67.73	45.41	70.52
Xie *et al*. [32]		15.03	8.70	15.82	27.34	19.21	28.35
Shin *et al*. [24]		*6.89*	4.99	*7.12*	*12.64*	10.47	*12.91*
Our diff. JPEG		**6.54**	**4.09**	**6.85**	**11.96**	**8.32**	**12.41**
Our diff. STE JPEG		7.22	*4.23*	7.59	13.16	*8.85*	13.69

Table 3. **IFGSM attack results summary.** Summarized top-1 and top-5 accuracy results after IFGSM attack for multiple JPEG quality ranges and different differentiable JPEG approaches. We report accuracy results for both $\epsilon = 3$ and $\epsilon = 9$.

Our approach w/o STE leads to particularly strong attack results for low JPEG qualities compared to other approaches (*cf*. Tab. 3). Our approaches (w/ & w/o STE) lead to strong adversarial images for a JPEG quality of 1, 2, and 3, while Shin *et al*. suffer to produce effective adversarial images (*cf*. Fig. 8). In general, our approach w/ STE leads to a stronger forward performance, while a better backward performance is achieved w/o STE.

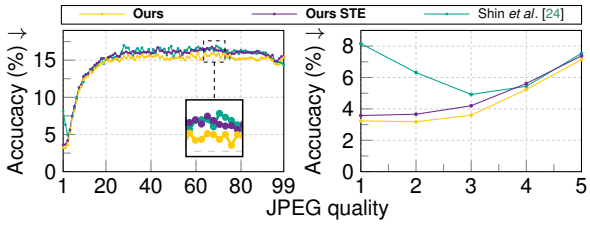

Figure 8. **IFGSM attack results.** Top-1 accuracy after IFGSM attack with our approach (w/ & w/o STE) *vs*. Shin *et al*. [24].

Vanishing gradient experiment. Both our differentiable JPEG w/ and w/o STE lead to lower gradient norms at the desired minima (*cf*. Tab. 4). The approach by Xing *et al*. leads to particularly large gradient norms. We suspect again the choice of rounding function approximation (Fourier-based) to be a major contributor to these results.

		$\mathbb{E}_{Q,\mathcal{D}}[\|\|\Delta_q \mathcal{L}_1\|\|]$			$\mathbb{E}_{Q,\mathcal{D}}[\|\|\Delta_{\mathbf{QT}_{Y,C}} \mathcal{L}_1\|\|]$		
Approach	q range →	1-99	1-10	11-99	1-99	1-10	10-99
Xing *et al*. [33]		1.254	7.271	0.502	–	–	–
Xie *et al*. [32]		0.955	4.617	0.492	0.913	1.041	0.896
Shin *et al*. [24]		0.587	4.172	0.139	0.213	0.474	0.181
Our diff. JPEG		*0.022*	**0.068**	*0.017*	*0.043*	**0.030**	*0.044*
Our diff. STE JPEG		**0.014**	0.077	**0.007**	**0.020**	*0.076*	**0.013**

Table 4. **Vanishing gradient results.** Average gradient norms at desired global minima. Gradient norms are averaged over the integer JPEG quality range Q and the Set14 [35] dataset $\mathcal{D}$.

6.3. Additional ablation results

Which rounding/floor approximation to use? In Tab. 5, we analyze the effect of different differentiable rounding and floor function approximations on the forward performance. Both the linear and the polynomial differentiable approximation perform best, with a slight advantage for the polynomial approach.

		SSIM ↑			PSNR ↑		
Function	q range →	1-99	1-10	11-99	1-99	1-10	10-99
Fourier $x - \sum_{k=1}^{10} \frac{(-1)^{k+1}}{k\pi} \sin(2\pi k x)$		0.984	0.956	0.987	41.28	35.66	41.98
Linear $\lfloor x \rceil + 0.1\,(x - \lfloor x \rceil)$		*0.990*	**0.987**	0.991	*42.37*	**38.70**	42.83
Polynomial $\lfloor x \rceil + (x - \lfloor x \rceil)^3$		**0.991**	*0.987*	**0.992**	**42.60**	*38.28*	**43.14**
Sigmoid $\sigma(60(x - \frac{1}{2} + \lfloor x \rfloor)) + \lfloor x \rfloor$		0.990	0.980	*0.992*	42.29	36.88	*42.96*
Tanh $\frac{1}{2}\tanh\big(5\,(x - \frac{1}{2} - \lfloor x \rfloor)\big) + \frac{1}{2}$		0.972	0.918	0.978	38.51	31.15	39.42

Table 5. **Rounding/floor ablation (forward function).** Performance of approximating the reference OpenCV [2] JPEG. For all experiments, our diff. JPEG w/o STE was used; only the differentiable rounding/floor approx. is varied. Eq. describe the rounding approximations; for the floor approx., we shift x by -0.5.

As in terms of backward function performance, the polynomial approximation also leads to the best adversarial examples (*cf*. Tab. 6). This suggests gradients derived from the polynomial approximation are more "useful" than from other approximations. While leading to a fair forward performance (*cf*. Tab. 5), the Fourier approximation leads to poor attack results, thus also yields worse gradients. This result aligns with the attack results by Xing *et al*. (*cf*. Tab. 2 & Tab. 3) also employing the Fourier approximation.

		Top-1 acc ↓			Top-5 acc ↓		
Function	q range →	1-99	1-10	11-99	1-99	1-10	10-99
Fourier		39.53	20.16	41.95	68.98	40.81	72.50
Linear		25.69	22.41	26.10	46.52	42.84	46.98
Polynomial		**14.39**	*7.97*	**15.19**	**25.79**	*17.53*	**26.83**
Sigmoid		*20.28*	**6.34**	*22.02*	*36.79*	**14.44**	*39.59*
Tanh		22.52	15.20	23.43	41.80	32.79	42.92

Table 6. **Rounding/floor ablation (IFGSM).** IFGSM attack results for various differentiable rounding/floor approximations. IFGSM w/ $\epsilon = 3$ used. For all experiments; our diff. JPEG was used, only the rounding/floor approximation was varied.

When considering both forward and backward performance (*cf*. Tab. 5 & Tab. 6), the best choice for approximating the rounding and floor function is the polynomial approach. These results might be explained by the fact the polynomial approximation strikes a vital trade-off between approximation error (w.r.t. the rounding/floor function), monotonicity, and gradient magnitude.

Which STE backward approximation to use? Standard STE assumes a constant gradient in the backward pass, while our STE approach uses a closer approximation of the true derivative. When using our proposed surrogate-based STE, we observe substantial gains in adversarial attack performance over standard STE (*cf*. Tab. 7), indicating that our approach leads to more "useful" gradients.

		Top-1 acc ↓			Top-5 acc ↓		
Backw. approach	q range →	1-99	1-10	11-99	1-99	1-10	10-99
Constant grad. (standard STE)		25.30	21.62	25.76	45.38	41.37	45.88
Surrogate (ours)		**7.22**	**4.23**	**7.59**	**13.16**	**8.85**	**13.69**

Table 7. **STE backward ablation (IFGSM).** IFGSM attack results for different STE backward approaches. IFGSM w/ $\epsilon = 3$ used. For all experiments, our differentiable JPEG w/ STE was used; only the STE backward approach was varied.

7. Conclusion

We reviewed existing differentiable JPEG approaches and proposed a novel differentiable JPEG approach modeling missing properties of standard (non-diff.) JPEG. Our approach is the first to accurately resemble standard JPEG over the entire JPEG quality range, outperforming all existing approaches. Notably, gradients derived from our approach yield superior adversarial examples, demonstrating the "usefulness" of the obtained gradients for gradient-based optimization. With strong forward and backward performance, our approach provides a solid foundation for future work considering JPEG as part of deep vision pipelines.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[2] Garys Bradski. The OpenCV Library. *Dr. Dobb's Journal*, 25(11):120–123, 2000. 1, 3, 6, 7, 8

[3] Jinyoung Choi and Bohyung Han. Task-Aware Quantization Network for JPEG Image Compression. In *ECCV*, pages 309–324, 2020. 1, 4

[4] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming Transformers for High-Resolution Image Synthesis. In *CVPR*, pages 12873–12883, 2021. 4

[5] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015. 5

[6] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. In *ICLR*, 2018. 1

[7] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster AutoAugment: Learning Augmentation Strategies using Backpropagation. In *ECCV*, pages 1–16, 2020. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 5

[9] Graham Hudson, Alain Léger, Birger Niss, István Sebestyén, and Jørgen Vaaben. JPEG-1 standard 25 years: past, present, and future reasons for a success. *Journal of Electronic Imaging*, 27(4):040901–1–040901–19, 2018. 1

[10] Abir Jaafar Hussain, Ali Al-Fayadh, and Naeem Radi. Image Compression Techniques: A Survey in Lossless and Lossy algorithms. *Neurocomputing*, 300:44–69, 2018. 2

[11] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017. 4

[12] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, et al. imgaug. https://github.com/aleju/imgaug, 2020. 1

[13] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data. In *NeurIPS*, volume 33, pages 12104–12114, 2020. 1

[14] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *ICLR*, 2017. 5

[15] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018. 5

[16] Xiyang Luo, Hossein Talebi, Feng Yang, Michael Elad, and Peyman Milanfar. The Rate-Distortion-Accuracy Tradeoff: JPEG Case Study. In *DCC*, pages 354–354, 2021. 1, 4

[17] David L MacAdam. Visual Sensitivities to Color Differences in Daylight. *Journal of the Optical Society of America*, 32(5):247–274, 1942. 2

[18] TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library. https://github.com/pytorch/vision, 2016. 6

[19] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021. 4

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, volume 32, 2019. 6

[21] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. In *WACV*, pages 3674–3683, 2020. 6

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 6

[23] Jian Shi, Edgar Riba, Dmytro Mishkin, Francesc Moreno, and Anguelos Nicolaou. Differentiable Data Augmentation with Kornia. *arXiv preprint arXiv:2011.09832*, 2020. 1

[24] Richard Shin and Dawn Song. JPEG-resistant Adversarial Images. In *NIPS Workshop on Machine Learning and Computer Security*, volume 1, page 8, 2017. 1, 4, 5, 6, 7, 8

[25] Connor Shorten and Taghi M Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *J. Big Data*, 6(1):1–48, 2019. 1

[26] Manli Shu, Yu Shen, Ming C Lin, and Tom Goldstein. Adversarial Differentiable Data Augmentation for Autonomous Systems. In *ICRA*, pages 14069–14075, 2021. 1

[27] Yannick Strümpler, Ren Yang, and Radu Timofte. Learning to Improve Image Compression without Changing the Standard Decoder. In *ECCVW*, pages 200–216, 2020. 1, 4

[28] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy Image Compression with Compressive Autoencoders. In *ICLR*, 2017. 4

[29] Aaron Van Den Oord, Oriol Vinyals, et al. Neural Discrete Representation Learning. *NIPS*, 30, 2017. 4

[30] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. 1, 2, 5

[31] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5

[32] Xiufeng Xie, Ning Zhou, Wentao Zhu, and Ji Liu. Bandwidth-Aware Adaptive Codec for DNN Inference Offloading in IoT. In *ECCV*, pages 88–104, 2022. 1, 4, 6, 7, 8

[33] Yazhou Xing, Zian Qian, and Qifeng Chen. Invertible Image Signal Processing. In *CVPR*, pages 6287–6296, 2021. 4, 6, 7, 8

[34] Yuankun Yang, Chenyue Liang, Hongyu He, Xiaoyu Cao, and Neil Zhenqiang Gong. FaceGuard: Proactive Deepfake Detection. *arXiv preprint arXiv:2109.05673*, 2021. 1

[35] Roman Zeyde, Michael Elad, and Matan Protter. On Single Image Scale-Up Using Sparse-Representations. In *Curves and Surfaces: 7th International Conference*, pages 711–730, 2012. 3, 6, 8

[36] Chaoning Zhang, Adil Karjauv, Philipp Benz, and In So Kweon. Towards Robust Data Hiding Against (JPEG) Compression: A Pseudo-Differentiable Deep Learning Approach. *arXiv preprint arXiv:2101.00973*, 2020. 1, 4

[37] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable Augmentation for Data-Efficient GAN Training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *NeurIPS*, volume 33, pages 7559–7570, 2020. 1

[38] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding Data With Deep Networks. In *ECCV*, pages 657–672, 2018. 1, 4