Deep Image Codec Control for Vision Models

Tiefe Bild Codec Regelung für Vision Modelle
Master thesis by Christoph Reich (Student ID: 2626466)
Date of submission: October 29, 2023

Supervisor: Prof. Stefan Roth, Ph.D. (VisInf Lab)
Supervisor: Oliver Hahn, M.Sc. (VisInf Lab)
External supervisor: Biplob Debnath, Ph.D. (NEC Labs)
Technische Universität Darmstadt, Darmstadt & NEC Laboratories America, Inc., Princeton

Deep Image Codec Control for Vision Models
Tiefe Bild Codec Regelung für Vision Modelle

Master thesis by Christoph Reich (Student ID: 2626466)

Date of submission: October 29, 2023

Technische Universität Darmstadt, Darmstadt & NEC Laboratories America, Inc., Princeton

Für den SV Bayer von 1904 und alle vergeudeten GPU Stunden,
ich habe ein absolut reines Gewissen!

Abstract

Standardized image coding builds the core of many image processing pipelines in the face of storage or network bandwidth requirements. However, standard codecs, such as JPEG, are optimized w.r.t. human quality assessment. The performance of downstream deep vision models is not considered by standard codecs. As deep vision models typically focus on silent parts or frequencies of an image, performing inference on JPEG-coded images leads to a vast deterioration in downstream performance. This thesis aims to develop a novel Deep Image Codec Control augmenting JPEG coding for deep vision models without breaking standardization. Instead of developing a novel image coded for deep vision models we augment JPEG, as standardization is required by the vast majority of real-world applications. Our codec control is trained in an end-to-end setting and predicts optimized codec parameters to preserve downstream performance and meet a given file size requirement. By considering the adherence to existing standardizations, downstream deep vision performance, and a dynamic target file size our Deep Image Codec Control is applicable to a wide range of real-world applications. To facilitate end-to-end learning we propose a novel differentiable JPEG coding approach. We also present a novel differentiable JPEG file size model, regressing the file size of the encoded JPEG image. This enables our Deep Image Codec Control learning to target a given file size condition in an end-to-end fashion. We showcase the feasibility of Deep Image Codec Control on three common computer vision tasks (image classification, object detection, and semantic segmentation). While only performing on par with existing approaches, our end-to-end learnable Deep Image Codec Control offers a novel and alternative direction for optimizing existing image codecs for deep vision models.

Note: Some parts of this thesis (Differentiable JPEG Coding, *c.f.* Sec. 4.1) have been accepted at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) as a full conference paper with the title ”Differentiable JPEG: The Devil is in the Details”. The paper will appear in the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision 2024 [1]. The papers project page is available at `https://christophreich1996.github.io/differentiable_jpeg/`. The proposed differentiable JPEG implementation is available at https://github.com/necla-ml/Diff-JPEG. The preprint of the paper is available at https://doi.org/10.48550/arXiv.2309.06978.

Zusammenfassung

Die standardisierte Bildcodierung bildet den Kern vieler Bildverarbeitungspipelines hinsichtlich Speicher- oder Netzwerkbandbreitenanforderungen. Standardcodecs wie JPEG sind jedoch auf die menschliche Qualitätsbewertung optimiert. Die Performance von Deep-Vision-Modellen wird von Standardcodecs nicht berücksichtigt. Da Deep-Vision-Modelle sich in der Regel auf einzelne Teilen oder Frequenzen eines Bildes fokussieren, führt dies zu einer erheblichen Verschlechterung der Performance, wenn JPEG-codierte Bilder von Deep-Vision-Modelle analysiert werden. Diese Arbeit zielt darauf ab, eine neuartige Tiefe Bild Codec Regelung zu entwickeln, die die JPEG-Codierung kontrolliert für Deep-Vision-Modelle, ohne dabei vorherrschende Standardisierungen zu brechen. Anstatt einen neuen Bildcodec für Deep-Vision-Modelle zu entwickeln, erweitern wir JPEG, da die Standardisierung für die überwiegende Mehrheit der Anwendungen essenziell ist. Unsere Codec-Steuerung wird in einem End-to-End-Setting trainiert und bestimmt optimierte Codec-Parameter, um die nachgelagerte Leistung zu erhalten und die vorgegebene Dateigröße einzuhalten. Durch Berücksichtigung der Einhaltung bestehender Standardisierungen, der Deep-Vision-Performance und einer dynamischen Ziel-Dateigröße ist unsere Tiefe Bild Codec Regelung in vielen realen Anwendungen einsetzbar. Um unsere Tiefe Bild Codec Regelung mittels End-to-End-Training zu lernen, präsentieren wir eine neue differenzierbare JPEG Implementierung. Wir stellen zusätzlich zu unserer differenzierbare JPEG Implementierung auch ein differenzierbares Modell zur Bestimmung der Dateigröße einer codierten JPEG-Datei vor. Dieses Modell ermöglicht es unsere Tiefe Bild Codec Regelung Codec Parameter zu bestimmen, welche zu einer vorgegebenen Dateigröße führen. Wir zeigen die Anwendung unserer Tiefe Bild Codec Regelung anhand von drei Standard Computer Vision Problemen (Bildklassifikation, Objekterkennung und semantische Segmentierung). Obwohl unsere Tiefe Bild Codec Regelung nur zu einer ähnlichen Performance bestehender Ansätze führt, bietet unsere End-to-End Tiefe Bild Codec Regelung eine neue und alternative Möglichkeit zur Optimierung bestehender Bildcodecs für Deep-Vision-Modelle.

Hinweis: Einige Teile dieser Arbeit (Differenzierbare JPEG-Codierung, vgl. Sec. 4.1) wurden als Konferenzveröffentlichung mit dem Titel ”Differentiable JPEG: The Devil is in the

Details" bei der IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) angenommen. Das Papier wird in den Proceedings der IEEE/CVF Winter Conference on Applications of Computer Vision 2024 veröffentlicht werden [1]. Die Projektseite des Papers ist unter folgender URL verfügbar: https://christophreich1996.github.io/differentiable_jpeg/. Die vorgestellte differenzierbare JPEG-Implementierung ist unter https://github.com/necla-ml/Diff-JPEG verfügbar. Der Preprint der Konferenzveröffentlichung is verfügbar unter https://doi.org/10.48550/arXiv.2309.06978.

Notations

This chapter provides an overview of the utilized notations in this thesis. Note this thesis mainly follows the notation of the Deep Learning book by Goodfellow *et al*. [2] and my previous thesis [3].

Numbers, Array, Tensors and Sets

a	A scalar
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathsf{\mathbf{A}}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columms
$\mathrm{diag}(\mathbf{a})$	A square, diagonal matrix with diagonal entries given by **a**
$\mathbb{R}$	Set of all real numbers
$\mathbb{R}^{\mathrm{H} \times \mathrm{W}}$	Set of real numbers over the dimensions H and W
$\mathbb{R}^{+}$	Set of all positive real numbers (without 0)
$\{0, 1, \ldots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b

Indexing

a_i	Element i of vector $\textsf{\textbf{a}}$, with indexing starting at 1
a_{-i}	All elements of vector $\textsf{\textbf{a}}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$\textsf{A}_{i,j,k}$	Element (i, j, k) of 3d tensor $\textsf{\textbf{A}}$
$\textsf{\textbf{A}}_{:,:,i}$	2d slice of 3d tensor $\textsf{\textbf{A}}$

Linear Algebra Operations

$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^+$	Moore-Penrose pseudoinverse of $\boldsymbol{A}$
$\boldsymbol{A} \odot \mathbf{B}$	Element-wise product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\det(\mathbf{A})$	Determinant of $\boldsymbol{A}$
$\boldsymbol{A} * \mathbf{K}$	Convolution operation between the matrices $\boldsymbol{A}$ and $\boldsymbol{K}$
$\boldsymbol{A} *^\top \mathbf{K}$	Transposed convolution between the matrices $\boldsymbol{A}$ and $\boldsymbol{K}$
$\mathrm{Tr}(\boldsymbol{X})$	Trace of the matrix $\boldsymbol{X}$

Calculus

$\frac{\mathrm{d}y}{\mathrm{d}x}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\textsf{\textbf{X}}} y$	Tensor containing derivatives y with respect to $\textsf{\textbf{X}}$

Probability

$\mathbb{E}[\boldsymbol{x}]$	Mean of the vector $\boldsymbol{x}$
$\mathbb{E}[f(x)]$	Expectation of the function $f(x)$
$\text{Var}[\boldsymbol{x}]$	Variance of the vector $\boldsymbol{x}$
$\text{Cov}[\boldsymbol{X}]$	Covariance of the matrix $\boldsymbol{X}$
$\text{Std}(\mathbf{X})$	Standard deviation of the tensor $\mathbf{X}$
$\mathcal{N}(\boldsymbol{x};\, \boldsymbol{\mu},\, \Sigma)$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance Σ
$\mathcal{N}(0, 1)$	Gaussian normal distribution with a mean of 0 and a variance of 1
$U(\boldsymbol{x};\, 0, 1)$	Uniform distribution over $\boldsymbol{x}$ in the interval $[0, 1]$
$U(0, 1)$	Uniform distribution with an interval of $[0, 1]$
$\text{KL}(P_1 \vert\vert P_2)$	Kullback-Leibler divergence of P_1 and P_2
$P(a)$	A probability distribution over a discrete variable
$p(a)$	A probability distribution over a continuous variable
$\mathbb{E}_{x \sim p}\left[f(x)\right]$	Expectation of $f(x)$ with respect to the distribution p

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f(\mathbf{X};\, \Theta)$	A function of $\mathbf{X}$ parametrized by Θ
$\log(x)$	Natural logarithm of $x \in \mathbb{R}$
$\exp(x)$	Exponential function of $x \in \mathbb{R}$
$\max(x,\, y)$	Maximum function of $x,\, y \in \mathbb{R}$

Abbreviations

ACC	accuracy.
BiLSTM	Bidirectional Long Short-Term Memory [4], [5].
BN	Batch Normalization [6].
CBN	Conditional Batch Normalization [7].
CNN	convolutional neural network.
DCT	discrete cosine transform.
DETR	Detection Transformer [8].
FGSM	Fast Gradient Sign Method [9].
GN	Group Normalization [10].
IFGSM	Iterative Fast Gradient Sign Method [11], [12].
MAE	mean absolute error.
mAP	mean Average Precision.
MARE	mean absolute relative error.
mIoU	mean Intersection-over-Union.
MSE	mean squared error.
PSNR	peak signal-to-noise ratio.
RLE	run-length encoding.
SSIM	structural similarity index measure.
STE	Straight-Through Estimator [13].
ViT	Vision Transformer [14].
XAI	explainable AI [15].
XCA	Cross-Covariance Attention [16].

Contents

1. Introduction

Lossy image compression is a necessity in the face of storage or network bandwidth requirements. Image, as well as video data, is a major and constantly increasing source of internet traffic. A significant and increasing amount of this image data is not consumed by humans but analyzed by deep learning-based vision models. However, standard lossy image codecs, such as JPEG [17], are optimized to trade compression strength against image distortion w.r.t. human quality assessment [18]. Downstream performance of deep vision models (*e.g.*, object detection performance) is not considered.

As deep vision models tend to focus on silent parts or frequencies of an image, the downstream performance of standard vision models vastly deteriorates as compression strength increases [19]–[25]. This behavior is showcased in Fig. 1.1 for three common computer vision tasks: semantic segmentation, object detection, and image classification. Otani *et al*. demonstrated a similar behavior for the downstream task of action recognition [23].

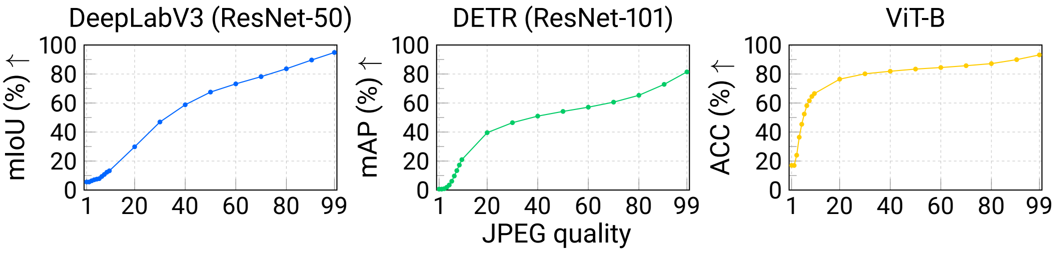


Figure 1.1. **Deep vision performance on JPEG-coded images.** We report downstream performance for three common vision tasks: semantic segmentation (DeepLabV3 [26]), object detection (DETR [8]), and image classification (ViT-B [14]). As the compression strength increases (lower JPEG quality), downstream performance vastly deteriorates. We use the predictions on the original images as pseudo labels. More details and additional experiments are provided in Sec. 2.2.

Standardized lossy image codecs are the *de facto* standard for the vast majority of applications requiring image compression [18]. While deep learning-based image codecs tend to outperform standard image codecs in benchmark settings, the most widely used lossy image coding approach remains JPEG [17], [18], [27]–[31]. Standard codecs offer better support for vastly different compression strengths, provide strong controllability over the encoding, and are computationally more efficient than deep image codecs. Additionally, standardization builds the basis of many applications. In contrast, no widely supported standard for deep image codecs has emerged yet, severely limiting the general applicability of deep image codecs [32].

This thesis aims to augment standard JPEG coding for deep vision models while adhering to existing standardizations. Given an image and target file size we want to learn a model predicting JPEG codec parameters (quantization tables) such that downstream deep vision performance is preserved. Additionally, the actual file size of the JPEG-encoded image should adhere to the target file size. We formulate this task as a constrained optimization problem [24].

$$\begin{aligned} \max_{\text{(codec parameters)}} \quad & \text{Deep Vision Model Performance} \\ & \text{s.t. Actual file size} \leq \text{Target file size.} \end{aligned} \tag{1.1}$$

We aim to learn a deep control network to solve this constrained optimization problem for different images and dynamic file size conditions. This thesis presents the first end-to-end learned deep image codec control for standard image codec while considering both downstream vision performance and a dynamic target file size. However, facilitating end-to-end learning of the deep image codec control is non-trivial since both JPEG encoding-decoding and the actual file size are non-differentiable.

To overcome the non-differentiability of JPEG encoding-decoding this thesis presents a novel differentiable JPEG coding approach. While various differentiable JPEG approaches have been proposed (*e.g.*, [33], [34]) we demonstrate that all existing approaches fail to approximate standard JPEG well when utilizing strong compression strengths (low JPEG quality). We show that existing differentiable JPEG approaches fail to model crucial operations of standard JPEG (*e.g.*, not considering discretizations of standard JPEG) and make suboptimal design choices (*e.g.*, poor rounding approximations). 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 [13] (STE) variant of our differentiable JPEG. Our differentiable JPEG approach 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. 1.2 while the approach by *et al*. [33] breaks down for a JPEG quality of 1 (strong compression) our differentiable JPEG approach offers a strong approximation over the while JPEG quality range. While we utilize our differentiable JPEG approach to optimize JPEG for deep vision models, possible applications of our approach range from differentiable data augmentation [35]–[41], data hiding [42], [43], and deep-fake detection [44] to adversarial attacks [33], [45].

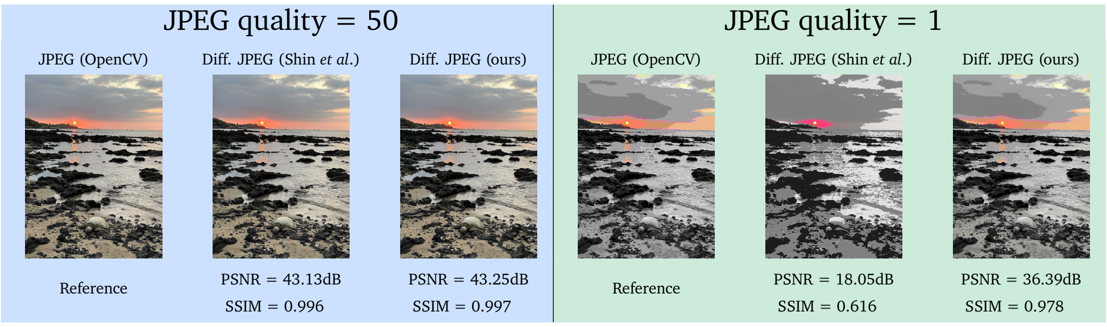
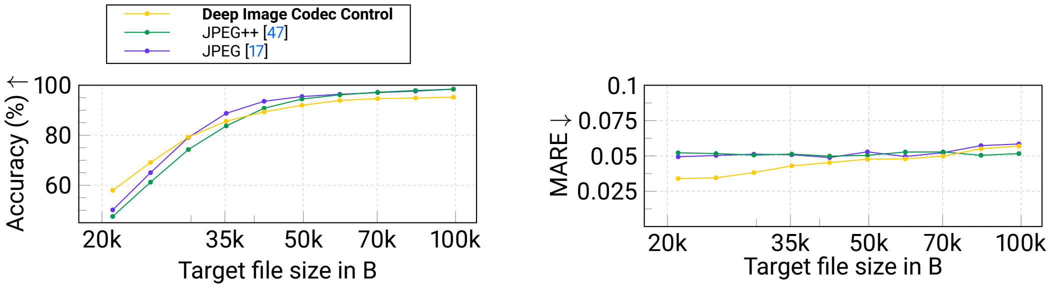


Figure 1.2. **Differentiable JPEG teaser results.** For a JPEG quality of 50, both Shin *et al*. [33] and our differentiable JPEG approach approximate the standard JPEG coding well. When reducing the JPEG quality to 1, the approach by Shin *et al*.does not approximate the JPEG coding well, while our 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 [46]).

To enable end-to-end learning w.r.t. the file size condition, we present a novel differentiable JPEG file size surrogate model. This surrogate model consumes the (differentiably) quantized DCT features and regresses the size of the JPEG-coded byte file. Our differentiable JPEG file size model enables us to derive gradient-based feedback w.r.t. the actual generated file size given an image and JPEG coding parameters.

With both our differentiable JPEG coding approach and JPEG file size surrogate model we can formulate Eq. (1.1) as an end-to-end learning problem. In particular, we end-to-end (self-supervised) learn a Deep Image Codec Control to solve our constrained optimization problem in a single forward pass. The core of our Deep Image Codec Control, the control network, consumes a file size condition and predicts JPEG quantization tables optimizing downstream deep vision performance while adhering to the given file size condition. By utilizing the downstream prediction on the original (not coded) image as a pseudo label our Deep Image Codec Control training does not require any annotations for training[1].

[1] Training the respective downstream model might require human annotations.



(a) Semantic segmentation results

(b) File size condition results

Figure 1.3. **Deep Image Codec Control teaser results.** Results of our Deep Image Codec Control (in yellow 🟨). Preservation semantic segmentation performance (a), measured by pixel-wise accuracy (ACC). Adherence to a dynamic file size condition (b), measured by mean absolute relative error (MARE). For reference, we also show the performance of our baselines (in purple 🟪 and green 🟩).

We showcase the application of our Deep Image Codec Control to three common computer vision tasks: image classification, object detection, and semantic segmentation. By learning a Deep Image Codec Control for a specific downstream task and model, we showcase the general application of our approach. While our approach only leads to on-par results with existing approaches in preserving downstream performance and adhering to a dynamic target file size (*c.f.* Fig. 1.3), we demonstrate the general feasibility of end-to-end learning to control JPEG for vision models. We outline difficulties in training our Deep Image Codec Control, such as training instability, and suggest future directions to remedy these difficulties.

1.1. Contributions

This thesis consists of five main contributions toward an end-to-end learnable Deep Image Codec Control. Our five main contributions are:

1. We thoroughly analyze the impact of JPEG coding on downstream deep vision performance. In particular, we show for three common downstream tasks that downstream deep vision performance vastly deteriorates when JPEG coding is utilized. For strong compression strengths, downstream performance for all tasks almost completely collapses.

2. We formulate the image codec control problem as a constrained optimization problem of preserving downstream deep vision performance and meeting a target file size. We present a relaxed loss-based training objective for learning a lightweight control network to optimize the constrained optimization objective in an end-to-end learning setting.

3. To enable a gradient flow w.r.t. the downstream deep vision models performance we propose a novel differentiable JPEG encoding-decoding approach. Our differentiable JPEG approach outperforms all existing differentiable JPEG methods.

4. We present a differentiable JPEG file size model, estimating the file size of the encoded image given both the image to be codec and codec parameters. This enables us to provide gradient-based feedback w.r.t. the generated file size.

5. We propose an end-to-end learnable Deep Image Codec Control for JPEG. We showcase the feasibility of our approach by showcasing the application of our Deep Control to three common vision tasks (image classification, semantic segmentation, and object detection).

2. Background

This chapter introduces the required fundamentals for this thesis and provides additional details on our motivation. In particular, we first review the JPEG compression standard before showcasing the downstream vision performance of various deep vision models on JPEG-coded images.

2.1. Background: The JPEG Coding Standard

The JPEG (Joint Photographic Experts Group) compression standard [17], [48], 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 (*c.f.* Sec. 2.1.3). Finally, the compressed JPEG file is produced using lossless

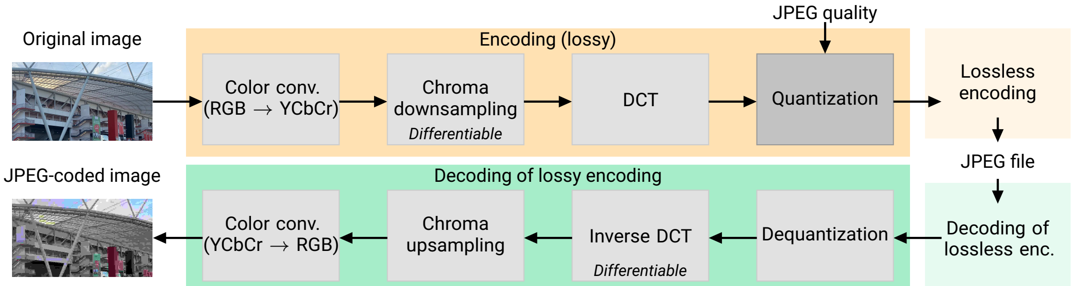


Figure 2.1. **The JPEG encoding-decoding pipeline.** The original input image is encoded to a JPEG file in a lossy manner. To recover the coded image the encoding is reversed in the decoding. JPEG uses lossless coding in conjunction with lossy coding.

coding. During decoding, the lossless and lossy encoding steps are reversed to reconstruct the JPEG-coded image from the JPEG file. Fig. 2.1 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 $\hat{\mathbf{I}}$

$$\begin{gathered}\mathrm{JPEG}\left(\mathbf{I}, q\right) = \hat{\mathbf{I}}, \quad q \in \{1, 2, \ldots, 99\} \\ \mathbf{I}, \hat{\mathbf{I}} \in \{1, \ldots, 255\}^{3 \times \mathrm{H} \times \mathrm{W}}.\end{gathered} \tag{2.1}$$

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.

2.1.1. 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 (*c.f.* Fig. 2.1). In the following, we explain the details of all encoding steps.

Color conversion (RGB $\to$ 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.

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cb} \\ \mathrm{Cr} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.2}$$

Chroma subsampling. The human eye tends to be more sensitive to variations in brightness than to color details [49]. 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 $\boldsymbol{p} \in \{0, 1, \ldots, 255\}^{64}$, the DCT is described by $\hat{\boldsymbol{p}} = \boldsymbol{a} \odot \boldsymbol{G}\boldsymbol{p}$. $\odot$ denotes the Hadamard product, $\boldsymbol{G} \in \mathbb{R}^{64 \times 64}$ contains the DCT coefficients, and $\boldsymbol{a}$ is a scaling factor. $\boldsymbol{G}$ is computed by $G_{8u+v,8i+j} = \cos\left(\frac{2x+1}{16}\right) \cos\left(\frac{2y+1}{16}\right)$ and $\boldsymbol{a}$ 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 $u, v, i, j \in \{0, 1, \ldots, 7\}$. $\hat{\boldsymbol{p}} \in \mathbb{R}^{64}$ 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:

$$s(q) = \begin{cases} \frac{5000}{q} & \text{if } q < 50 \\ 200 - 2q & \text{otherwise.} \end{cases} \tag{2.3}$$

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

2.1.2. JPEG decoding

The JPEG decoding converts the compressed binary JPEG file back to an RGB image (*c.f.* Fig. 2.1). 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 (*c.f.* Fig. 2.1).

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.

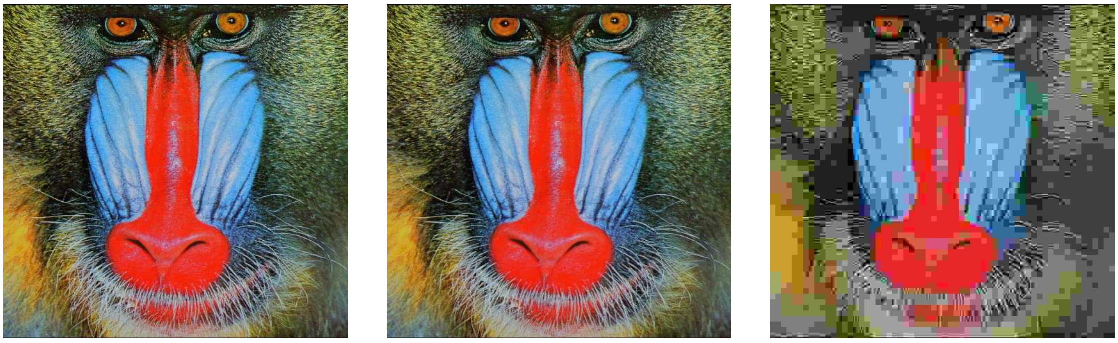
Dequantization. To dequantize, the quantized DCT features are multiplied with the respective scaled QT (luma or chroma table) $\tilde{\boldsymbol{P}} = \overline{\boldsymbol{P}} \odot \hat{\boldsymbol{QT}}$. For simplicity, we omit the distinction between the luma and chroma channels.

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 (*c.f.* Eq. (2.2))

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \left(\begin{bmatrix} \mathrm{Y} \\ \mathrm{Cb} \\ \mathrm{Cr} \end{bmatrix} - \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.344136 & -0.714136 \\ 1 & 1.772 & 0 \end{bmatrix} . \tag{2.4}$$



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

Figure 2.2. **JPEG coding artifacts.** (a) Original image, (b) JPEG-coded image with a JPEG quality of 50, file size is 47.3kB (bottom image), and (b) coded image with a JPEG quality of 1, file size is 6.2kB (bottom image). Images from the Set14 [50] and OpenCV [46] JPEG used.

2.1.3. 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 (*c.f.* Fig. A.1b) since quantization suppresses plenty of frequencies. *Vice versa*, a high JPEG quality leads to a larger file size but reduces distortion (*c.f.* Fig. 2.2b). This tradeoff is known as the rate-distortion trade-off.

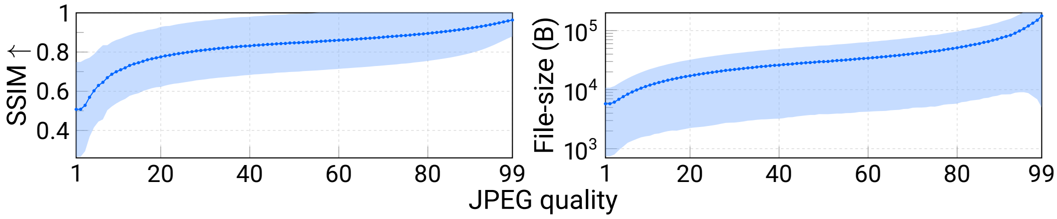


Figure 2.3. **Rate-distortion trade-off.** JPEG rate-distortion trade-off curve on the Set14 [50] dataset. We report both the mean and two standard deviations.

JPEG employs the DCT in a patch-wise manner to achieve efficient compression. This compression approach introduces distinctive artifacts in the resulting 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 (*c.f.* Fig. 2.2b).

2.1.4. Non-differentiability of JPEG

The JPEG encoding-decoding process (*c.f.* Eq. (2.1)) 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 ($\mathrm{JPEG}_c : \mathbb{R}^{3\times\mathrm{H}\times\mathrm{W}} \times [1, 99] \rightarrow \mathbb{R}^{3\times\mathrm{H}\times\mathrm{W}}$). 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.

2.2. Deep Vision Performance on JPEG-Coded Images

While strong compression leads to increased image distortion, compression strength also affects downstream deep vision performance. This behavior is also known as the rate-distortion-accuracy trade-off [47]. In this section, we demonstrate empirically how JPEG coding affects the performance of different deep vision models. In particular, we showcase that all tested models struggle to maintain downstream performance. Our results on three classical computer vision tasks: image classification (*c.f.* Sec. 2.2.1), object detection (*c.f.* Sec. 2.2.2), and semantic segmentation (*c.f.* Sec. 2.2.3), align with the findings by Otani *et al*. [23] and Luo *et al*.[47]. Otani *et al*. observed that JPEG coding can vastly deteriorate action recognition performance, similarly Luo *et al*. showed the same behavior for the task of image classification.

2.2.1. Image classification

We tested different image classification networks on JPEG-coded ImageNet-1k [51] (*c.f.* Sec. 5.1.3) samples. In particular, we utilize three ResNet [52] variants and three Vision Transformer [14] (ViT) [14] variants from timm [53] trained on ImageNet-1k. For different JPEG quality values, we report the top-1 and top-5 accuracy w.r.t. the $\arg\max$ prediction on the original images (*c.f.* Fig. 2.4).

Fig. 2.4 showcases the image classification results. All models lose downstream performance as stronger compression (lower JPEG quality) is utilized. We observe that, in particular, smaller models struggle to maintain performance, whereas larger model variants tend to be a bit more robust against JPEG coding.

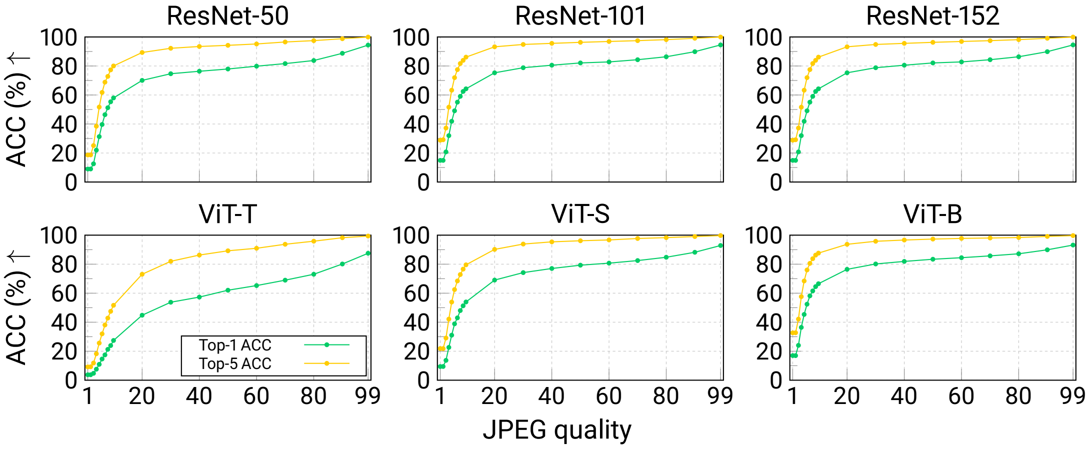


Figure 2.4. **Image classification performance on JPEG-coded images.** Performance of all models vastly deteriorates as compression strength increases (lower JPEG quality). We employ the image classification predictions obtained on the original images as pseudo labels. We report the top-1 ACC in green 🟩 and the top-5 ACC in yellow 🟨.

2.2.2. Object detection

For the downstream task of object detection, we utilize the Detection Transformer [8] (DETR) model with different backbone networks (ResNet-50 & ResNet-101) on the COCO 2017 [54] dataset (*c.f.* Sec. 5.1.3). For the full integer range of JPEG quality values, we perform validation of the DETR models. We employ the prediction on the original (non-coded) image as a pseudo label. In particular, we use the arg max class prediction and bounding box prediction on the original image. We use the mean Average Precision (mAP) metric to validate the object detection performance.

Consistent with the image classification results (*c.f.* Fig. 2.4), object detection performance vastly deteriorates as the compression strength increases (lower JPEG quality), as shown in Fig. 2.5. Notably, the object detection performance deteriorates more evenly than image classification. Similar to image classification a larger backbone network also leads to slightly more robust object detections on JPEG-coded images.

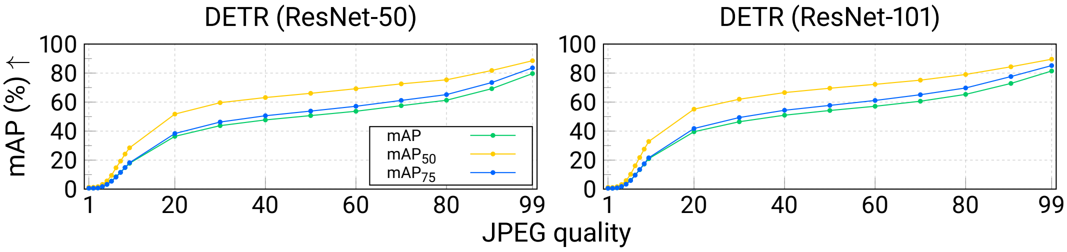


Figure 2.5. **Object detection performance on JPEG-coded images.** Performance of both DETR [8] models vastly deteriorates as compression strength increases (lower JPEG quality). We employ the object detection predictions obtained on the original images as pseudo labels. We report the mAP in green 🟩, the mAP_{50} in yellow 🟨, and the mAP_{75} in blue 🟦.

2.2.3. Semantic segmentation

In the case of semantic segmentation, we utilize the common DeepLabV3 [26] model with different ResNet variants (ResNet-50 & ResNet-101) as backbones. Both variants are trained on the Cityscapes [55] dataset and are provided by MMSegmentation [56]. We measure the mean Intersection-over-Union (mIoU) between the prediction on the coded image and the pseudo label (arg max prediction) on the original image.

As both in image classification and object detection, semantic segmentation performance suffers when using a strong compression strength (low JPEG quality). This is showcased in Fig. 2.6. Surprisingly, semantic segmentation downstream performance deteriorates even more evenly over the JPEG quality range.

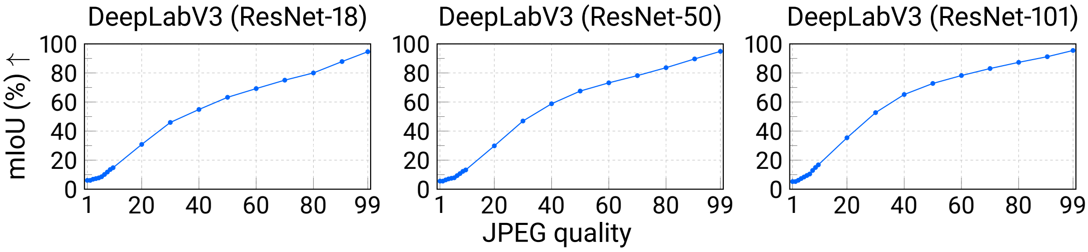


Figure 2.6. **Semantic segmentation performance on JPEG-coded images.** Performance of both models vastly deteriorates as compression strength increases (lower JPEG quality). We employ the semantic segmentation predictions obtained on the original images as pseudo labels. We report the mIoU (in blue 🟦).

2.2.4. Discussion: JPEG data augmentation

To introduce robustness against photometric noise and other transformations (*e.g.*, flipping or Gaussian noise) data augmentations are typically employed during training deep vision model [39], [57], [58]. A naive approach for introducing robustness against JPEG distortion would be to utilize JPEG coding as a data augmentation approach. This naive approach entails two major limitations. First, it would be required to train downstream models again from scratch using JPEG coding as a data augmentation technique[1]. Second, Otani *et al*. [23] showed that using JPEG during training can only slightly introduce robustness against JPEG-coded images. This behavior has also been showcased for H.264 coded videos on the task of semantic segmentation [24]. Due to these limitations, this thesis proposes an alternative approach for approaching this problem that is orthogonal to using data augmentation. We aim to augment/control JPEG such that only information irrelevant to the downstream vision performance is discarded and performance is better preserved than using standard JPEG. Note that our approach does not require any changes to the downstream deep vision model.

[1] Or a sophisticated fine-tuning approach would be required.

3. Related Work

This section reviews existing work in the scope of our Deep Image Codec Control. In particular, we first introduce existing differentiable JPEG coding approaches and methods for modeling the JPEG file size in a differentiable setting. Finally, we review existing work on optimizing coding for deep vision models.

3.1. Differentiable JPEG Coding

STE-based approaches. In standard STE, the gradient of a non-differentiable function is approximated by assuming a constant gradient of one [13]. During the forward pass, the true non-differentiable function is used. STE has been shown to be effective in various deep learning-based approaches [13], [59]–[62]. This technique is also used when the gradient of a function would be zero a.e. (*e.g.*, rounding function). Choi *et al.* [63] used STE to propagate gradients through the rounding operation of the JPEG encoding [64]. InSTEad of assuming a constant gradient as in standard STE, Xie *et al.* [65] 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, clipping, or flooring) 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 [33]. 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 $s(q) = \begin{cases} \frac{50}{q} & \text{if } q < 50 \\ 2 - \frac{2q}{100} & \text{otherwise} \end{cases}$ and $\hat{\boldsymbol{QT}} = s\boldsymbol{QT}_\text{S}$. Note this leads to a difference of 0.5 in the scale quantization table $\hat{\boldsymbol{QT}}$ compared to the standard scaling factor computation (*c.f.* Eq. (2.3)), subsequently deteriorating the approximation performance. Other approaches have been built on the approach by Shin *et al*. with slight modifications [34], [47], [66]. 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 (*c.f.* Sec. 2.1.3 and Fig. 2.2). This motivates noise-based approaches, wherein JPEG coding is approximated by introducing specific noise into the original image. Zhu *et al*. [43] 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*. [42] 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.

3.2. Differentiable JPEG File Size Modeling

Estimating the file size given both an image and JPEG coding parameters is no-differentiable as lossless utilizes discreet operation. Additionally, measuring the required bytes is also no-differentiable. Further, the encoded JPEG file size is non-trivially dependent on both the image content and the coding parameters. For instance, an entirely black image requires vastly fewer bytes than a natural image encoded by the same JPEG coding parameters. This behavior is qualitatively showcased in Fig. 3.1 and also causes the large standard deviation in Fig. 2.3 (right).

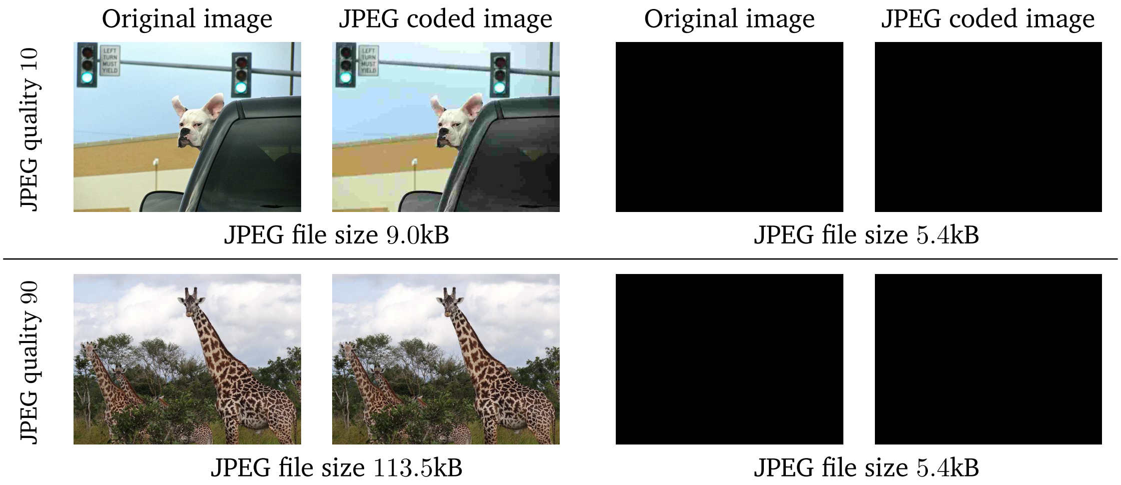


Figure 3.1. **JPEG file size example.** JPEG file size difference between a natural image from the COCO dataset [67] and an entirely black image (same resolution). For encoding we used the same encoding parameters (standard quantization tables used). While encoding both images with the same encoding parameters the JPEG file size vastly differs.

To still provide gradient-based feedback w.r.t. the generated JPEG file size multiple differentiable approximations of the JPEG file size have been introduced [47], [63], [65]. On a high level, lossless JPEG encoding can be interpreted as a variant of entropy coding [17]. To this end, Luo *et al*. [47] proposed an entropy approximation approach. To approximate the true probability mass function a three-layer feed-forward neural network is employed. To mimic the actual JPEG lossless coding two separate feed-forward neural networks are used, one for the luma channel and one for the chroma channels. Additionally, two feed-forward networks are utilized for estimating the density function of zero frequency and none-zero frequencies, respectively. While this entropy estimator-based approach can approximate the JPEG file size in a differentiable manner the estimated file size tends to overshoot the actual file size, especially for larger bit-rates [47]. Xie *et al*. [65] proposes an STE-based approach for approximating the JPEG run-length encoding. While this approach closely resembles run-length encoding, JPEG Huffman encoding is not considered. This, especially, leads to a deterioration in the approximation of the JPEG file size for high-resolution images. Choi *et al*. [63] modeled both the run-length and Huffman encoding with a separate neural network. For modeling the run-length encoding symbol prediction a Bidirectional Long Short-Term Memory [4], [5] (BiLSTM) is employed. To achieve the final discrete symbol prediction the Gumbel-Softmax reparameterization trick is applied [60]. However, the use of the Gumbel-Softmax reparameterization trick introduces stochasticity to the file size prediction. For approximating the Huffman encoding two feed-forward neural networks are employed for the luma channel and the chroma channels, respectively.

In contrast to Xie *et al*. [65], we model both the run-length and JPEG Huffman encoding. Instead of using a two-stage approach with Gumbel-Softmax reparameterization as Choi *et al*. [63], we utilize a single powerful network for regressing the JPEG file size. This enables the prediction of the JPEG file size without stochasticity.

3.3. Optimizing Image Coding for Vision Models

As deep vision models preserve imagery differently than humans research has been devoted to optimizing image coding for deep vision models. This is also referred to as image coding for machines. Deep image codecs offer support for custom objectives. Subsequently, significant afford have been invested to learn deep image codecs for downstream deep vision model [68]–[76]. Deep image codec approaches for machines can be broadly clustered into feature-based (*e.g.*, [68], [69]) and image-focused approaches (*e.g.*, [71], [72]). Image-focused approaches follow the standard encoding-decoding approach of standard image codecs and recover a codec image. In contrast, feature-based image coding approaches focus on compressing feature representations that can be used after decoding for making downstream predictions. While deep image codecs offer the simple integration of custom objectives (*e.g.*, for optimizing downstream performance), can be used to compress feature representations, and typically reach a better rate-distortion trade-off their applicability is very limited in real-world settings. Compared to standard lossy image codecs, deep image codes do not offer standardization as no standard has emerged yet. Additionally, deep image codecs are significantly more computationally demanding and only offer limited control over the encoding. Reaching a specific file size during inference is typically not supported. For a more detailed discussion on the limitations and capabilities of deep image coding, we refer the reader to the respective survey paper [29], [77], [78].

Another line of research has focused on optimizing standard codecs, such as JPEG, for machines [24], [47], [63], [65], [79]–[82]. JPEG, as well as other standard codecs, offer support for customizations of the encoding [17], [18], [83]. In particular, JPEG offers support for custom quantization tables, controlling the quantization strength (*c.f.* Sec. 2.1). This has motivated recent approaches to optimize JPEG w.r.t. the encoding parameters [47], [63], [65], [80]. More specifically, most approaches aim to optimize the JPEG quantization tables, as the quantization tables offer a strong control of the encoding process [17], [47]. GRACE [80] utilizes gradient feedback from a downstream model to analyze the sensitivity to different DCT frequency bands of a downstream model to optimize the quantization strength, no file size conditioning is supported. Luo *et al*. [47] propose an approach for trading rate (file size) *vs*. image distortion and downstream accuracy. By using continuous gradient-based optimization w.r.t. a loss-based criterion, trading rate *vs*. distortion and accuracy, optimized quantization tables have been proposed. While the proposed quantization tables maximize both distortion and accuracy the estimated quantization tables are fixed and can not adapt to different images. A specific target file size during inference is also not considered. Xie *et al*. [65] presented an approach for optimizing the quantization tables, the color conversation parameters (RGB → YCbCr), and downstream deep vision model weight w.r.t. to downstream performance and a target file size. This approach also only considers fixed quantization tables during inference and only targets a single file size condition during training. No dynamic conditioning during inference is supported. Similar to Luo *et al*., Xie *et al*.also only estimates quantization tables that are fixed during inference. Instead of employing fixed quantization tables during inference, Choi *et al*. [63] learns a convolutional neural network (CNN) to predict optimized quantization tables for a given input image (dynamic quantization tables). The CNN is trained to predict optimized quantization tables w.r.t. both downstream deep vision performance and a specific target file size. However, during inference, the learn CNN can only generate dynamic quantization tables targeting the, during training, utilize file size condition. To target different file sizes the CNN needs to be retrained which is impractical when targeting a large range of different file sizes.

In contrast to existing approaches, we want to learn a control network predicting dynamic quantization tables and being able to adapt to different file size conditions during inference. In particular, given an input image and a target file size our control network learns to solve a constrained optimization problem w.r.t. downstream deep vision performance and a target file size (*c.f.* Eq. (1.1)). A high-level compression of our novel Deep Image Codec Control and existing approaches is provided in Tab. 3.1.

Approach	Optimize deep vision performance	Dynamic file size condition	Dynamic Quantization tables
Luo *et al.* [47]	✓	✗†	✗
Xie *et al.* [65]	✓	✗*	✗
Choi *et al.* [63]	✓	✗*	✓
Ours	✓	✓	✓

* denotes the support of a (fixed) file size condition during training time

† rate-distortion-accuracy tradeoff can be achieved by loss weighting (no direct target file size)

Table 3.1. **Overview of related work.** A high-level overview of existing approaches in comparison to our Deep Image Codec Control.

4. Method

In this section, we introduce our Deep Image Codec Control methodology. First, we introduce a novel differentiable JPEG coding approach to enable a gradient propagation from the coded JPEG image to the JPEG parameters (*e.g.*, quantization tables). Then we introduce an approach for modeling the JPEG file size in a differentiable setting. This allows us to compute a gradient-based signal for the generated file size. Finally, we introduce our Deep Image Codec Control pipeline for deep vision models.

4.1. Differentiable JPEG Coding

We aim to build a continuous and differentiable approximation

$$\text{JPEG}_{\text{diff}}:[0,255]^{3\times\mathrm{H}\times\mathrm{W}}\times[1,99]\rightarrow[0,255]^{3\times\mathrm{H}\times\mathrm{W}} \tag{4.1}$$

of the full JPEG encoding-decoding function (*c.f.* Eq. (2.1)). This approximation should accurately resemble standard (non-differentiable) JPEG and yield gradients "useful" for gradient-based optimization. To achieve this, we first take a surrogate model approach (Sec. 4.1.1). Later we present the incorporation of the STE technique (Sec. 4.1.2).

Note our differentiable JPEG function can easily adapted to accept the quantization tables ($\boldsymbol{QT}_Y$ and $\boldsymbol{QT}_C$) as inputs $\text{JPEG}_{\text{diff}}\left(\mathbf{I},\boldsymbol{QT}_Y,\boldsymbol{QT}_C\right)$. This is achieved by omitting the scaling by the JPEG quality.

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

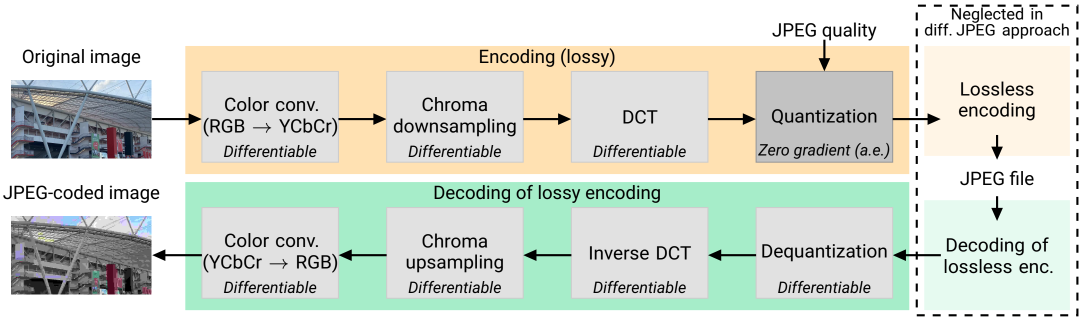


Figure 4.1. **The JPEG differentiable encoding-decoding pipeline.** We follow the standard JPEG coding pipeline and extend all differentiable operations with their naive continuous generalization. Non-differentiable operations are replaced with a differentiable surrogate. Since no information is lost (identity mapping) during the lossless encoding/decoding we can neglect these coding steps in our differentiable JPEG approach.

operations. Note for operations not described (*e.g.*, DCT), we use their naive continuous generalization. Since the gradient is not affected by the lossless encoding/decoding (*c.f.* Fig. 2.1), we follow existing approaches and neglect these steps.

4.1.1. Differentiable JPEG surrogate

Differentiable quantization. For approximating the quantization operation, which uses the rounding function, we utilize the polynomial approximation

$$\lfloor x \rceil + (x - \lfloor x \rceil)^3 \tag{4.2}$$

by Shin *et al*. [33]. While other approximations exist (*e.g.*, sigmoid function), we later show that this design choice leads to superior performance over other approximations.

Differentiable QT scale floor. Non-differentiable standard JPEG computes the quantization table scaling $s(q)$ based on the JPEG quality with integer arithmetic (*c.f.* Eq. (2.3)). 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 (*c.f.* Eq. (2.3)) is used, not the approach by Shin *et al*. [33].

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

$$\lfloor x - 0.5 \rceil + (x - 0.5 - \lfloor x - 0.5 \rceil)^3. \tag{4.3}$$

We later validate this design choice against other approximations.

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 [17], the quantization table is bounded to the integer range $\{1, 2, \dots, 255\}^{8\times 8}$. 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 $\overline{\text{clip}}$.

$$\overline{\text{clip}}(x) = \begin{cases} x & \text{if } x \in [b_{\text{min}}, b_{\text{max}}] \\ \gamma\, x & \text{otherwise} \end{cases}, \; x \in \mathbb{R}, \gamma \in (0, 1]. \tag{4.4}$$

This soft approximation ensures a non-zero gradient of x when outside of the range $[b_{\text{min}}, b_{\text{max}}]$. 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 $\{0, 1, \dots, 255\}$. 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.

4.1.2. 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.

4.2. Differentiable JPEG File Size Modeling

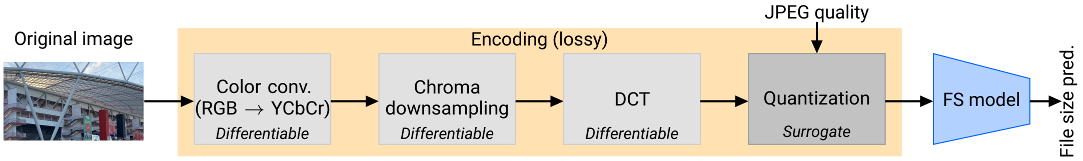


Figure 4.2. **Differentiable JPEG file size pipeline.** We utilize the differentiable JPEG coding encoder (*c.f.* Sec. 4.1) to approximate the lossy encoding. Based on the output of the diff. lossy encoding we learn a differentiable JPEG file size model (in blue 🟦) regressing the JPEG file size.

We aim to build a differentiable estimator of the JPEG file size. While there are differentiable upper bounds for estimating the file size of entropy coding available, we learn a differentiable JPEG file size model regressing the file size of the JPEG byte code (*c.f.* Fig. 4.2). This JPEG file size model makes use of our differentiable JPEG (lossy) encoder and models the compression of run-length encoding (RLE) and Huffman encoding. In particular, the file size model takes in the quantized DCT features of the differentiable (lossy) encoder and predicts the file size. Note this is, in particular, a challenging task since the file size depends both on the image content and the utilized JPEG parameters (JPEG quality and quantization tables). For instance, an entirely black image in a vastly smaller file size than a natural image, both encoded with the same coding parameters.

4.2.1. Differentiable JPEG file size model

We view the prediction of the JPEG file size based on the quantized DCT features. In particular, the JPEG file size model takes in the patch-wise (luma and chroma) features as a sequence of tokens and regresses the file size (*c.f.* Fig. 4.3). To cope with high-resolution images we utilize a XCiT-like model (*c.f.* Fig. 4.3) for this regression task [16]. As the number of tokens can vastly increase as resolution increases using a standard transformer can be infeasible [16], [84]. Cross-Covariance Attention [16] (XCA) solves the quadratic memory complexity during training the standard transformer by building a per channel attention matrix instead of a per token one [16].

Our differentiable JPEG file size model is composed of four XCiT blocks [16] followed by two class attention blocks [85]. The full architecture is shown in Fig. 4.3. The differentiable JPEG file size model takes in the quantized DCT features of both the luma and chroma channels. Before feeding the tokens into the first XCiT block we contacted sinusoidal positional encodings and linearly embed each token. All tokens are processed by the four XCiT blocks. For making our file size prediction we draw inspiration from the Class Attention approach [85]. We utilize two Class Attention blocks processing both the quantized DCT tokens and a learnable regression token. After the two Class Attention blocks, we extract the transformed regression token and utilize a prediction head (linear layer & Softplus activation [86]) to predict the file size of the encoded JPEG file.

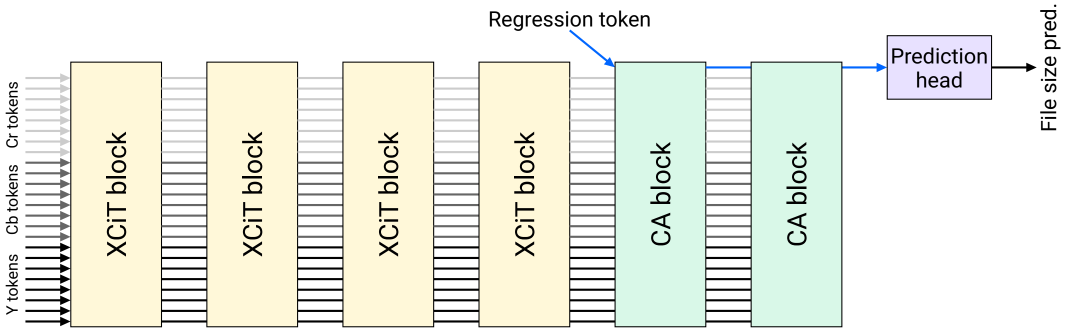


Figure 4.3. **Differentiable JPEG file size model architecture.** Our JPEG file size model is composed of four XCiT blocks [16] (in yellow 🟨), two Class Attention blocks [85] (in green 🟩) and a prediction head (in purple 🟪) for regressing the file size from a learnable regression token (in blue 🟦). For the sake of similarity we omit the incorporation of positional encodings and the linear input mapping.

The XCiT block (*c.f.* Fig. 4.4) processes a sequence of N tokens $\boldsymbol{T} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ with an embedding dimension D. Our XCiT block is composed of two Layer Normalization operations [87], a two-layer feed-forward network (with GELU activation [88]), and a XCA layer. The XCA follows the standard transformer by utilizing linear input projections $\boldsymbol{Q} = \boldsymbol{W}_{\boldsymbol{Q}}\boldsymbol{T}$, $\boldsymbol{K} = \boldsymbol{W}_{\boldsymbol{K}}\boldsymbol{T}$, and $\boldsymbol{V} = \boldsymbol{W}_{\boldsymbol{V}}\boldsymbol{T}$. However, instead of performing attention over the token dimension, XCA performs attention over the embedding dimension D by

$$\text{XC-Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \boldsymbol{V}\, \text{Softmax}\left(\hat{\boldsymbol{K}}^{\mathsf{T}}\hat{\boldsymbol{Q}}\right). \tag{4.5}$$

Note that $\hat{K}$ and $\hat{Q}$ denote the normalized query and key matrix. XCA utilizes the Euclidean norm to normalize the key and query matrix. Again similar to the standard transformer a linear output projection is utilized. Intuitively, the XC-Attention operation can be interpreted as a dynamic 1×1 convolution, where the convolutional weight is constructed dynamically based on the given input features. Similar to the standard transformer XC-Attention also utilizes multiple attention heads to enhance expressiveness. This yields an operation linear in memory and computational complexity w.r.t. the number of input tokens. For reference, the standard attention formulation can be viewed as a dynamic linear layer, leading to a memory (only during training [89]) and computational quadratic complexity w.r.t. the number of input tokens.

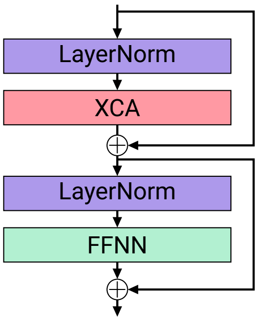


Figure 4.4. **Modified XCiT building block.** Our XCiT-based building block is composed of two Layer Normalizations in purple 🟪, a XCA layer in red 🟥, and a channel-wise feed forward neural network (FFNN) in green 🟩.

Note that our differentiable JPEG file size model can be easily integrated into our differentiable JPEG coding surrogate. This, subsequently, requires only a single forward pass through the differentiable (lossy) encoding.

4.2.2. Differentiable JPEG file size training

We are interested the our differentiable JPEG file size model learns to regress the JPEG file but also provides accurate gradient approximations. In particular, we formulate the JPEG file size mapping as a black box function

$$\begin{gathered}\mathrm{JPEG}_{\mathrm{fs}} = \mathrm{sizeof}\left(\mathrm{JPEG}_{\mathrm{e}}\left(\mathbf{I}, q\right)\right) = f, \quad q \in \{1, 2, \ldots, 99\} \\ \mathbf{I} \in \{1, \ldots, 255\}^{3 \times \mathrm{H} \times \mathrm{W}}, \quad f \in \mathbb{N}^{+},\end{gathered} \tag{4.6}$$

with $\mathrm{JPEG_e}$ denoting the JPEG encoding and sizeof representing the operation of querying the file size (in bytes) from the JPEG binary file. Similar, to the differentiable JPEG coding approach (*c.f.*, Sec. 4.1) we can generalize this function to arbitrary quantization tables $(\mathrm{sizeof}\left(\mathrm{JPEG_e}\left(\mathbf{I}, \boldsymbol{QT}_Y, \boldsymbol{QT}_C\right)\right) = f)$.

To learn a low-variance gradient estimator of Eq. (4.6) we make use of the control variates theory. In particular, our JPEG file size model can become a low-variance gradient estimator of the JPEG file size mapping if *(1)* the error between the output of the differentiable JPEG file size model and the true function is minimized and *(2)* the two output distributions are maximizing the correlation coefficient ρ [90]. Both requirements can be enforced during training.

To minimize the error we utilize a MARE loss defined as

$$\mathcal{L}_{\text{MARE}} = \frac{1}{n}\sum_{i=1}^{\text{B}} \frac{\left|f_i - \hat{f}_i\right|}{|f_i|}. \tag{4.7}$$

Where $\hat{f}_i$ denotes the (indexed) batched prediction of the file size and f_i represents the (indexed) batched true file size. B is the utilized batch size. The relative error is utilized to equally weight the different file sizes, independent of their magnitude. While it's difficult to learn a neural network over multiple orders of magnitude we regress the file size in $\log_{10}$-space. For maximizing the correlation coefficient ρ we can use a correlation coefficient loss [91]

$$\mathcal{L}_{\rho} = \frac{\sum_{k=1}^{\text{B}}\left(f_k - \mathbb{E}[\boldsymbol{f}]\right)\left(\hat{f}_k - \mathbb{E}\left[\hat{\boldsymbol{f}}\right]\right)}{\sqrt{\sum_{k=1}^{\text{B}}\left(f_k - \mathbb{E}[\boldsymbol{f}]\right)^2}\sqrt{\sum_{k=1}^{\text{B}}\left(\hat{f}_k - \mathbb{E}\left[\hat{\boldsymbol{f}}\right]\right)^2}}.$$

Note this loss function requires a batch size B strictly larger than one. During training we minimize a linear combination of both loss functions $\mathcal{L}_{\text{fs}} = \mathcal{L}_{\text{MARE}} + \lambda_\rho\mathcal{L}_\rho$. $\lambda_\rho \in \mathbb{R}^+$ denotes a positive scalar factor applied to the correlation coefficient loss.

We pre-train and validate our differentiable JPEG model on the respective downstream datasets (*c.f.* Sec. 5.1) by randomly sampling quantization tables from $U(1, 255)$ and minimizing $\mathcal{L}_{\text{fs}}$. To adapt to the quantization tables predicted by our control network, we fine tune the differentiable JPEG file size model in a lazy fashion during the end-to-end coded control training (*c.f.* Algorithm 1). We perform lazy training, *i.e.* updating the differentiable JPEG file size model only every n-th training step, since training requires a forward pass through Eq. (4.6) which is comparably slow since JPEG coding (w/ custom quantization tables) is performed on the CPU with Pillow [92].

4.3. Deep Image Codec Control

We aim to learn a lightweight control network (*c.f.* Fig. 4.5) predicting JPEG codec parameters (*i.e.*, the quantization tables) solving the constrained optimization problem

$$\begin{aligned} \max_{(\boldsymbol{QT}_Y, \boldsymbol{QT}_C)} \quad & \mathrm{M}(\mathrm{DNN}(\mathrm{JPEG}(\mathbf{I}, \mathrm{C}(\mathbf{I}, f)))) \\ \text{s.t. } & \tilde{f} \leq f. \end{aligned} \tag{4.8}$$

Where $\mathbf{I}$ denotes the image to be coded, f the target file size, $\tilde{f}$ the actual file size of the encoded image, and JPEG the JPEG encoding-decoding mapping. The main objective is the maximization of a downstream metric M (*e.g.*, ACC) based on the prediction of a downstream deep neural network DNN (*e.g.*, ResNet-50). The optimization problem is constrained by the target file size f. In practice we want the actual file size to be close to the target file size. We consider the JPEG quantization tables $\boldsymbol{QT}_Y \in [1, 255]^{8\times8}$ and $\boldsymbol{QT}_C \in [1, 255]^{8\times8}$ as the predicted codec parameters of our control network C. Note using custom quantization tables does not break the standardizations of JPEG.

In simple words, we want to learn a control network, consuming both the image to be coded and a target file size, to predict codec parameters (quantization tables). These codec parameters should maximize the downstream performance of a deep vision model while leading to a file size within the target file size. Intuitively, we want to learn a control network that is able to solve the constrained optimization problem (*c.f.* Eq. (4.8)) in a single forward pass. Our full Deep Image Codec Control pipeline is showcased in Fig. 4.5.

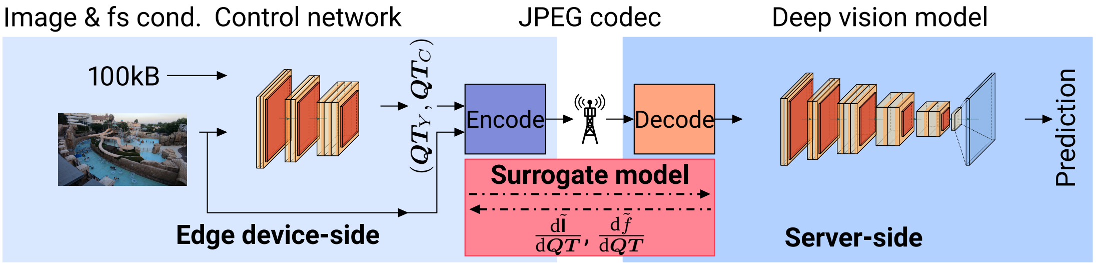


Image & fs cond.
Control network
JPEG codec
Deep vision model
100kB
$(\boldsymbol{QT}_Y, \boldsymbol{QT}_C)$
Encode
Decode
Surrogate model
$\frac{\mathrm{d}\tilde{\mathbf{I}}}{\mathrm{d}\boldsymbol{QT}}, \frac{\mathrm{d}\tilde{f}}{\mathrm{d}\boldsymbol{QT}}$
Edge device-side
Server-side
Prediction

Figure 4.5. **Deep Image Codec Control pipeline.** Given a file size condition and the image to be coded, the control network (*c.f.* Fig. 4.6) predicts the JPEG quantization tables. The image is encoded using the predicted quantization tables before being transferred over a network or being stored. The encoded image is decoded and analyzed by a deep vision model on the server-side. To enable training the control network in an end-to-end fashion a differentiable surrogate model of JPEG is used during training. Input image is taken from the REDS dataset [93].

4.3.1. Reinforcement learning discussion

We want to approach the learning of the control network by leveraging standard end-to-end learning. While learning the prediction of the quantization tables can also be approached by reinforcement learning [94], we argue that using reinforcement learning is infeasible to approach this problem due to the large action space of the quantization tables and the complex loss surface of the deep vision model. In particular, when only limited compute is available, stabilizing reinforcement learning over a large range of file size conditions and large datasets is infeasible.

4.3.2. End-to-end control training

While our constrained optimization problem (*c.f.* Eq. (4.8)) is non-differentiable due to the JPEG coding and the file size conditioning, we can utilize the proposed differentiable JPEG coding approach (*c.f.* Sec. 4.1) and our differentiable JPEG file size model (*c.f.* Sec. 4.2) to overcome this non-differentiability. In particular, we reformulate the constrained optimization problem as a loss-based minimization problem

$$\mathcal{L}_{\mathrm{control}} = \lambda_{\mathrm{c}}\,\mathcal{L}_{\mathrm{c}} + \lambda_{\mathrm{p}}\mathcal{L}_{\mathrm{p}} + \lambda_{\mathrm{r}}\,\mathcal{L}_{\mathrm{r}} + \lambda_{\mathrm{QT}}\,\mathcal{L}_{\mathrm{QT}}, \tag{4.9}$$

where $\mathcal{L}_{\mathrm{c}}$ is the file size condition loss, $\mathcal{L}_{\mathrm{p}}$ the performance loss, $\mathcal{L}_{\mathrm{r}}$ a file size regularizer, and $\mathcal{L}_{\mathrm{QT}}$ regularizes the quantization tables. λ_{c}, λ_{p}, λ_{r}, and λ_{QT} are positive weighting factors of the different loss components. Note that this general formulation is inspired by the recent work of Reich *et al.* [24].

Our codec control loss $\mathcal{L}_{\mathrm{control}}$ can be seen as a soft and continuous approximation of the constrained optimization problem (*c.f.* Eq. (4.8)). More specifically, our reformulation into a loss-based minimization problem (excluding regularizers) is equivalent to the Lagrangian function (w/ L1 loss on file size condition) of our constrained optimization problem. Intuitively, the file size condition loss $\mathcal{L}_{\mathrm{c}}$ penalizes the control network if the predicted codec parameters lead to exceeding the file size condition. $\mathcal{L}_{\mathrm{p}}$, the performance loss, provides a learning signal w.r.t. the performance of the downstream deep vision model. The file size regularizer $\mathcal{L}_{\mathrm{r}}$ enforces that the generated file size is close to the target file size. This is also beneficial for the downstream performance as we have shown that a smaller file size (stronger compression) leads to a deterioration in downstream performance (*c.f.* Sec. 2.2). The quantization table regularizer $\mathcal{L}_{\mathrm{QT}}$ constrains the space of possible solutions, leading to a more stable and faster convergence.

File size condition loss. For enforcing our file size conditioning, we employ a loss between the predicted (differentiable) JPEG file size $\tilde{f}$ and the file size condition f

$$\mathcal{L}_{\mathrm{c}} = \max(0, \tilde{f} - f\,(1 - \delta_{\mathrm{c}})). \tag{4.10}$$

This loss formulation can be seen as a one-sided L1 loss, penalizing if the file size condition f is exceeded. δ_{c} is a zero or positive constant pushing $\tilde{f}$ below the file size condition f. Note, due to the large numerical range of the file size condition we apply $\mathcal{L}_{\mathrm{c}}$ in $\log_{10}$-space to circumvent exploding gradients.

Performance loss. To ensure downstream performance is maximized we utilize a performance loss minimizing a criterion w.r.t. the downstream performance. The performance loss is dependent on the utilized downstream task. In the case of image classification, we utilize a cross-entropy loss defined as

$$\mathcal{L}_{\mathrm{p}} = -\sum_{j=1}^{\mathrm{C}} \tilde{p}_j \log \hat{p}_j, \tag{4.11}$$

where $\tilde{\boldsymbol{p}}$ denotes the argmax thresholded one-hot classification over the classes C obtained on the original (raw) image. We also refer to $\tilde{\boldsymbol{p}}$ as the pseudo label. $\hat{\boldsymbol{p}}$ represents the softmax prediction obtained on the coded image.

For the downstream task of semantic segmentation, we utilize the multi-class focal loss [95] defined as

$$\mathcal{L}_{\mathrm{p}} = \frac{1}{\mathrm{HW}} \sum_{i=1,\, l=1}^{\mathrm{H,W}} \sum_{j=1}^{\mathrm{C}} -\alpha\,(1 - \hat{\mathsf{P}}_{j,i,l})^{\gamma} \tilde{\mathsf{P}}_{j,i,l} \log \hat{\mathsf{P}}_{j,i,l}. \tag{4.12}$$

$\tilde{\mathbf{P}} \in 0, 1^{\mathrm{C}\times\mathrm{H}\times\mathrm{W}}$ denotes the thresholded pseudo semantic segmentation label obtained on the original (non-coded) image and $\hat{\mathbf{P}} \in [0, 1]^{\mathrm{C}\times\mathrm{H}\times\mathrm{W}}$ indicates the softmax semantic segmentation on the coded image over the semantic classes C. H and W denote the spatial dimensions. α denotes a scalar weighting factor and γ the focusing parameter. We set both to the default values 0.5 and 2, respectively. We utilize the focal loss over a standard cross-entropy loss to down-weight well-classified pixels as, especially, for a high target file size the semantic segmentation prediction is relatively close to the pseudo label.

In the case of object detection (w/ DETR), we compute the performance loss both on the object classifications and the bounding box predictions. In particular, we compute

$$\mathcal{L}_{\mathrm{p}} = \sum_{i=1}^{\mathrm{N}} \Big[\underbrace{-\sum_{j=1}^{\mathrm{C}} \tilde{P}_{i,j} \log \hat{P}_{i,j}}_{\mathcal{L}_{\mathrm{oc}}} + \underbrace{\beta \; \mathbb{1}_{\arg\max \tilde{P}_i \neq \varnothing} \Big|\Big| \tilde{B}_i - \hat{B}_i \Big|\Big|_1}_{\mathcal{L}_{\mathrm{box}}} \Big], \tag{4.13}$$

where $\mathcal{L}_{\mathrm{oc}}$ is the object classification loss which computes the cross-entropy between the thresholded object classification pseudo label $\tilde{\boldsymbol{P}} \in [0,1]^{\mathrm{N}\times\mathrm{C}}$ obtained on the original image and the object classification $\hat{\boldsymbol{P}} \in [0,1]^{\mathrm{N}\times\mathrm{C}}$ on the coded image. Note that N denotes the object classification predictions of DETR and C represents the semantic classes. $\mathcal{L}_{\mathrm{box}}$ is the bounding box loss enforcing that the bounding box prediction on the original image $\tilde{\boldsymbol{B}} \in \mathbb{R}^{\mathrm{N}\times 4}$ (pseudo label) and the predictions on the coded image are matching $\hat{\boldsymbol{B}} \in \mathbb{R}^{\mathrm{N}\times 4}$. Note we only compute the absolute error between bounding boxes if not the no object class $\varnothing$ is predicted by $\tilde{\boldsymbol{P}}$, this is denoted by $\mathbb{1}_{\arg\max \tilde{P}_i \neq \varnothing}$. β is a positive scalar value weighting the bounding box loss. We do not match predictions with the pseudo label, as the original DETR training does since we want to precisely match the prediction obtained on the original image. Note, that using a different task than the presented ones might require adapting the performance loss $\mathcal{L}_{\mathrm{p}}$ to the respective downstream task.

File size regularizer. To enforce the control network to predict quantization tables leading to file size close but below the target file size, we utilize a file size regularizer. Our file size regularizer is defined as

$$\mathcal{L}_{\mathrm{r}} = \left| \min(0, \tilde{f} - f\,(1 - \delta_{\mathrm{r}})) \right|, \tag{4.14}$$

where $\tilde{f}$ again denotes the predicted (differentiable) JPEG file size and f the file size condition. δ_{r} is used to not push the file size above the file size condition. In general, we set δ_{r} such that $\delta_{\mathrm{r}} \geq \delta_{\mathrm{c}}$. Similar to the file size loss $\mathcal{L}_{\mathrm{c}}$, we also compute the file size regularizer $\mathcal{L}_{\mathrm{r}}$ in $\log_{10}$-space to compensate for the large numerical range.

Quantization table regularizer. The space of possible quantization tables optimizing our constrained optimization problem (*c.f.* Eq. (4.8)) is enormous. Hence, end-to-end training is unstable especially when a small batch size is used. To constrain the space of possible quantization table solutions we regularize the quantization table prediction. In particular, we constrain the average standard deviation of the predicted quantization tables by

$$\mathcal{L}_{\mathrm{QT}} = \left| \frac{1}{\mathrm{B}} \left(\sum_{i=1}^{\mathrm{B}} \mathrm{Std}(\boldsymbol{QT}_{Y,i}) \right) - \zeta_{\mathrm{QT}} \right| + \left| \frac{1}{\mathrm{B}} \left(\sum_{i=1}^{\mathrm{B}} \mathrm{Std}(\boldsymbol{QT}_{C,i}) \right) - \zeta_{\mathrm{QT}} \right|, \tag{4.15}$$

where ζ_{QT} denotes the target standard deviation and B the batch size. We refer to Luo *et al*. [47] for finding a suitable standard deviation target. In particular, Luo *et al*. [47] proposed static but optimized quantization tables for deep vision models, we utilize these quantization tables to determine ζ_{QT}.

Algorithm 1 End-to-end image codec control training. PyTorch-like [96] pseudo code of the proposed self-supervised image codec control training approach.

```
for image in data_loader:
   # Sample a file size condition
   fs_c = log_uniform(bw_min, bw_max)
   # File size condition into log10-space
   fs_c_log = log10(fs_c)
   # Forward pass control network
   qt_y, qt_c = control_network(image, fs_c_log)
   # Forward pass of surrogate model
   image_jpeg, fs_p_log = diff_jpeg(image, qt_y, qt_c)
   # Prediction on coded image
   pred = downstream_model(image)
   # Generate pseudo label with uncompressed clip
   with no_grad():
      pred_pseudo = downstream_model(clip)
      label_pseudo = threshold_pred(label_pseudo)
   # Compute control loss
   loss_control = lambda_c * loss_c(fs_p_log, fs_c_log) \
      + lambda_p * loss_p(pred, label_pseudo) \
      + lambda_r * regularizer_fs(fs_p_log, fs_c_log) \
      + lambda_qt * regularizer_qt(qt_y, qt_c, target_std)
   # Compute backward pass and perform optimization
   loss_control.backward()
   optimizer_control_network.step()
   # Next: Make lazy JPEG file size model training step
```

Training procedure. We train the control network and the differentiable JPGE file size model in an alternating fashion. In particular, we train the control network and for every

n-th step we train the differentiable JPEG file size model in a lazy fashion. The whole training procedure is showcased in Algorithm 1. Note we only use our differentiable JPEG approach during training. For validation we utilize the Pillow JPEG implementation with support for custom quantization tables [92].

Bootstrap training

The control network is required to learn a dynamic behavior over a wide range of different file size conditions. This can lead to instability during the early stages of the training. To compensate for this we bootstrap the control network on the optimized quantization tables proposed by Luo *et al*. [47]. In particular, we sample a random JPEG quality value to scale the optimized quantization tables. JPEG encoding is performed with the scale quantization tables to obtain the true JPEG file size. The true JPEG file size is then fed into the control network to obtain the quantization table prediction of the control network. The predicted quantization tables are then supervised by an absolute error to match the (scaled) optimized quantization tables. With this approach we provide a clear target for the control network, subsequently stabilizing the control network training during the start of the training. We also optimize the control network w.r.t. the control loss $\mathcal{L}_{\text{control}}$ (*c.f.* Eq. (4.9)) in an alternating fashion.

Note that during preliminary testing we occasionally observed a divergence of the end-to-end control training when not using bootstrapping. When using bootstrapping no divergence has been observed.

4.3.3. Connections to knowledge distillation and XAI approaches

While we propose our self-supervised training approach in light of controlling JPEG, our training approach can also be interpreted under the lens of knowledge distillation [97]. We aim to learn quantization tables w.r.t. both downstream performance and a file size condition. To this end, we take gradient-based feedback from the downstream deep vision model. This can also be seen as an approach of knowledge distillation, as we want to extract from the downstream model which frequencies are important to preserve w.r.t. downstream performance. By taking gradient feedback from the downstream deep vision model we distill knowledge from the downstream model to the control network.

We can further draw connections to explainable AI [15] (XAI) approaches. Given an image or video, compute vision XAI methods typically want to infer pixel or region-wise attribution maps explaining the importance of features for a downstream prediction (*e.g.*, image classification) [19], [22], [98]. Methods like Grad-CAM [20], expected gradients [99], or integrated gradients [19], [22] extract gradient-based information from a deep vision model to infer interpretable explanations within the image space. Similarly, we want to inform our control network about the importance of specific features in the DCT space w.r.t. a downstream prediction. However, different from XAI approaches we do not use gradient-based information during inference but we want to train the control network to understand the feature importance of unseen images. Additionally, instead of predicting pixel or region-wise importance maps we want to estimate feature importance in the frequency space (DCT space) in the form of JPEG quantization tables.

4.3.4. Control network

Our control network consumes both the image to be coded as well as the file size condition and predicts the quantization tables ($\boldsymbol{QT}_Y$ & $\boldsymbol{QT}_C$, codec parameters). We design our control network with the requirement of achieving a very lightweight architecture applicable also in constrained resource environments (*e.g.*, phones or edge devices). In particular, our control network is composed of a standard backbone and a conditional prediction head (*c.f.* Fig. 4.6). As the backbone, we employ a very lightweight ConvNeXt model in the atto configuration, pre-trained on ImageNet-1k [53], [100]. The conditional prediction head is composed of three conditional residual blocks, consuming both the latent features from the backbone, DCT statistics of the input image as well as the file size condition, and predicting the codec parameters. To ensure a correct spatial dimension (matching the quantization table size) we downsample the spatial dimension of the backbone features to a spatial resolution of 8×8 After the final conditional residual block, we use a linear mapping followed by a sigmoid activation. We scale the output by 255 to scale the prediction to the value range of the quantization tables (the value range is $[1, 255]$). The whole control network architecture is depicted in Fig. 4.6.

As most CNNs (including ConvNeXt [100]) do not adhere to the Nyquist Shannon Theorem [101] features obtained from a CNN backbone might not capture all frequency components of a given input image [102], [103]. To still inform the prediction head about all present frequency components we concatenate DCT statistics of the input image to the backbone features. In particular, we compute the channel-wise first and second-order statistics (mean and variance) of the patch-wise DCT features from the input image in the YCbCr color space. We concatenate the channel-wise statistics with the backbone features before feeding both to the conditional prediction head.

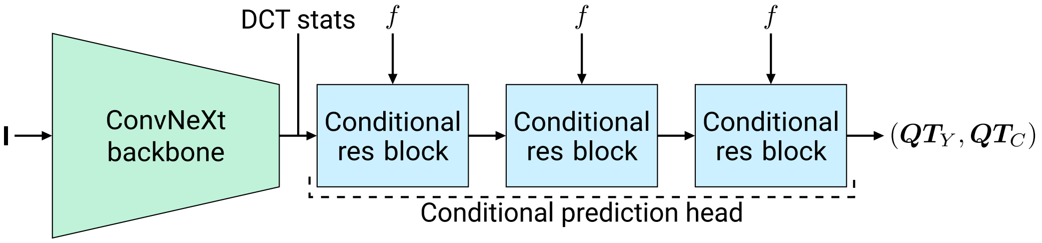


Figure 4.6. **Control network architecture.** Visualization of the control networks architecture. The control network consumes the image to be coded $\mathbf{I}$ and the target file size f as a condition and predicts the optimal JPEG quantization tables ($\boldsymbol{QT}_Y$ & $\boldsymbol{QT}_C$). We utilize a ConvNeXt (atto) [53], [100] (pre-trained on ImageNet-1k) to extract image features (in green 🟩). Three conditional residual blocks (in light blue 🟦) are used to incorporate the file size condition (conditional prediction head) and predict the quantization tables. We feed both the features of the ConvNeXt backbone and DCT statistics into the conditional prediction head.

The conditional residual block used by the conditional prediction head is showcased in Fig. 4.7. By utilizing Conditional Batch Normalization [7] (CBN) the file size condition (in $\log_{10}$-space) is incorporated into the prediction. Inspired by self-supervised learning approaches, we utilize Batch Normalization [6] (BN) in the CBN layer to counteract a collapse of the predicted quantization tables to a single prediction [104], [105]. During preliminary experiments, we observe a more stable training with BN than using, for instance, Group Normalization [10] (GN).

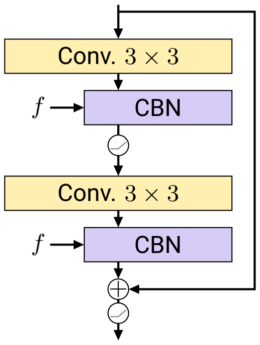
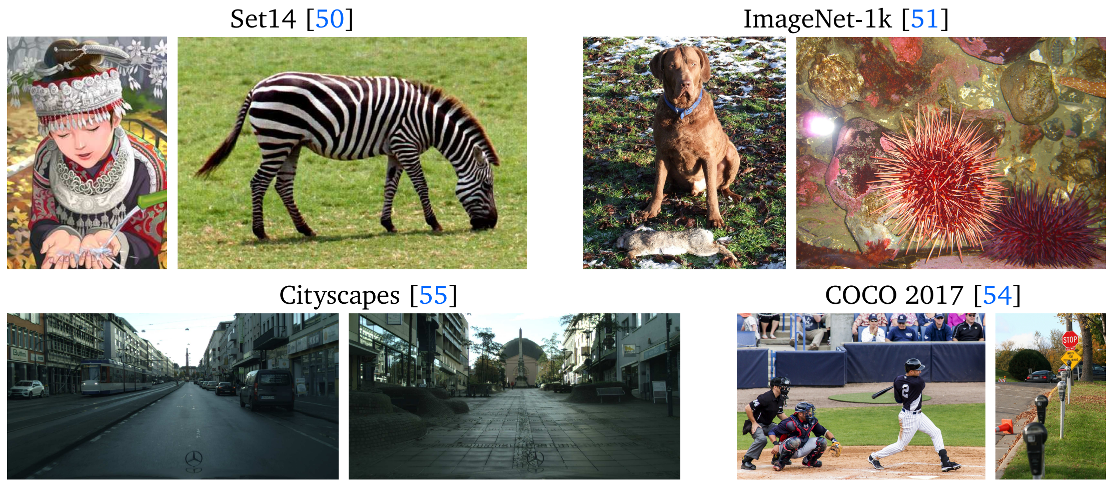


Figure 4.7. **Conditional residual building block.** Visualization of the conditional residual building block used in the conditional prediction head of our control network. Two convolutional layer (in yellow 🟨), two non-linear activations, a standard BN layer (in purple 🟪), and a CBN layer (in light purple 🟪) are used.

5. Experiments

This chapter provides empirical results of the proposed methodologies. First, we describe the dataset and evaluation metrics used. Then we introduce the utilized baseline followed by implementation details. Finally, we provide both results for our differentiable JPEF approach, our differentiable JPEG file size model, and our Deep Image Codec Control.

5.1. Datasets



Set14 [50]

ImageNet-1k [51]

Cityscapes [55]

COCO 2017 [54]

Figure 5.1. **Datasets overview.** Samples from the four diverse dataset used in our experiments.

This section describes the different datasets used for training and validating the proposed approaches. On a high level, for our differentiable JPEG coding experiments, we use ImageNet-1k [51] and the Set14 dataset [50]. For pre-training our differentiable JPEG file size model, we use video sequences of the Cityscapes dataset [55]. For our Deep Image

Codec Control experiments, dependent on the downstream task, we use the Cityscapes dataset [55], the ImageNet-1k dataset [51], or the COCO 2017 dataset [54]. Samples of the utilized dataset are depicted in Fig. 5.1.

5.1.1. Differentiable JPEG coding experiments

For our adversarial attack experiments, we utilize 5k randomly chosen images of the ImageNet-1k validation set (ILSVRC 2012) [51]. For all other experiments, we use the Set14 dataset [50], composed of 14 RGB images ranging from natural to document-like images.

5.1.2. Differentiable JPEG file size experiments

For pre-training our differentiable JPEG file size model we use the images of the unlabeled sequences of Cityscapes including high-resolution (1024×2048) 30-frame video clips (~ 10k frames total) with 2967 training and 498 validation clips. We validate the pre-trained model on the respective validation frames. We pre-train different differentiable JPEG file size models with the respective downstream resolution (*e.g.*, 224×224 for image classification). Note fine-tuning during Deep Image Codec Control training is performed on the respective downstream dataset.

5.1.3. Deep Image Codec Control experiments

Depending on the downstream task we use different datasets for training our Deep Image Codec Control. For image classification we utilize the ImageNet-1k [51] dataset, composed of 1.3M training and 50k images. We utilize a resolution of 224×224. For the task of semantic segmentation, we utilize the Cityscapes dataset [55] at a square resolution of 768×768, achieved via clipping. Cityscapes offers both a labeled set as well as an unlabeled set composed of short video sequences. For training, we utilize the 3k unlabeled video sequences of Cityscape as our approach does not require annotations. For validation, we use the 500 validation images from the labeled set. For object detection experiments we use the COCO 2017 dataset [54]. COCO is composed of 118k training and 5k validation samples at a resolution of 640×480, achieved via resampling.

5.1.4. Dataset discussion

All utilized datasets use some degree of compression. While there are approaches to remove compression artifacts (*e.g.*, JPEG coding artifacts), these approaches typically require knowledge of the applied compression [106]. Due to the use of various sources (*e.g.*, images from the internet) it is typically unknown which compression is applied to individual images. To compensate for this, we estimate the working range of JPEG on the respective dataset. Based on the rate-distortion curves, shown in Fig. 5.2 we determine the JPEG working range for each dataset used for our Deep Image Codec Control experiments.

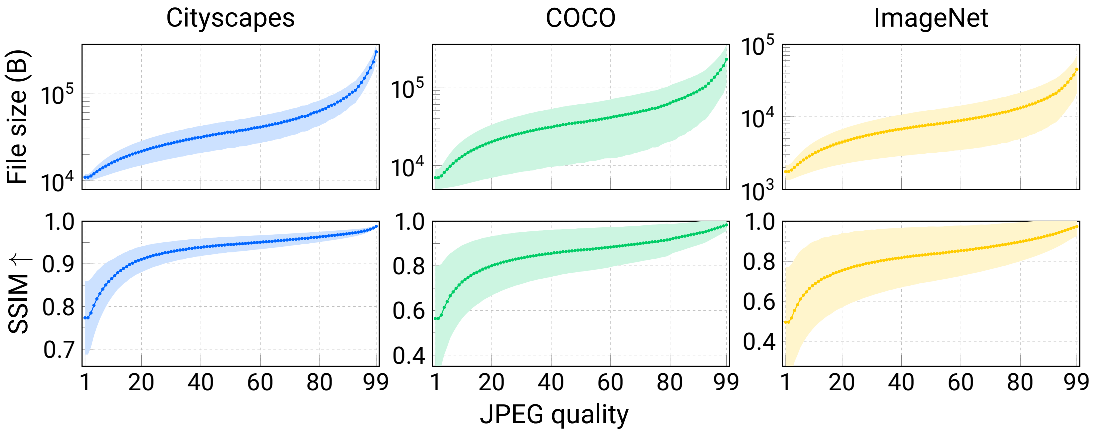


Figure 5.2. **JPEG rate-distortion curves.** JPEG rate-distortion curves for three different datasets: Cityscapes [55], COCO 2017 [54], and ImageNet-1k [51]. We utilize the respective validation set. Note that file sizes between datasets differ substantially due to the different image resolutions. We show the mean and two standard deviations.

5.2. Validation

5.2.1. Differentiable JPEG coding experiments

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 [107], the PSNR for evaluation, the mean squared error (MSE), and the mean absolute error (MAE) (L1).

Backward function evaluation. We aim to showcase the "usefulness" of gradients obtained by the backward function. Taking inspiration from Shin *et al.* [33] 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 $C\left(\text{JPEG}_{\text{diff}}\left(\mathbf{I}, q\right)\right) = \mathbf{p}$, composed of an ImageNet classifier C (*e.g.*, ResNet-50 [52]) and a differentiable JPEG function $\text{JPEG}_{\text{diff}}$, conditioned on a given JPEG quality q. We aim to craft an adversarial image $\mathbf{I}_{\text{adv}}$, 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 [9] (FGSM) [9] and the Iterative Fast Gradient Sign Method [11], [12] (IFGSM) [11], [12]. FGSM in the non-targeted setting crafts an adversarial example by

$$\mathbf{I}_{\text{adv}} = \mathbf{I} + \epsilon \ \text{sign}(\Delta_{\mathbf{I}}[\mathcal{L}(y, C\left(\text{JPEG}_{\text{diff}}\left(\mathbf{I}, q\right)\right))]), \tag{5.1}$$

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 $\mathcal{L}_1(\text{JPEG}_{\text{diff}}(\mathbf{I}, q), \text{JPEG}_{\text{r}}(\mathbf{I}, q))$. Subsequently, estimate the gradient norms w.r.t.to both the JPEG quality $\|\Delta_q \mathcal{L}_1\|$ and the (internal) standard QT $\|\Delta_{\boldsymbol{QT}_{Y,C}} \mathcal{L}_1\|$ (luma and chroma table). We average the gradient norms over both the different integer JPEG quality ranges and the dataset $\mathcal{D}$.

5.2.2. Differentiable JPEG file size experiments

We validate our differentiable JPEG file size model on how well the actual file size can be regressed. In particular, we measure the MARE. We perform validation after pre-training and sample quantization tables randomly from a uniform distribution $U(1, 255)$. Note, that similar to the differentiable JPEG coding evaluation, validating the backward pass of our differentiable JPEG file size model is not defined. However, due to the control variates theory we can assume that our differentiable JPEG file size model becomes a low-variance gradient estimator if the file size is regressed accurately [90], [108]. The Deep Image Codec Control also provides a cue if our differentiable JPEG file size model provides an accurate gradient estimate. Since no accurate gradients are generated our Deep Image Codec Control could not learn to predict parameters w.r.t. target file size.

5.2.3. Deep Image Codec Control experiments

We validate our Deep Image Codec Control for ten different file size conditions within the JPEG working range (*c.f.* Fig. 2.3). For each file size condition, we perform one validation pass over the respective validation set. We report both a task-specific performance metric (*e.g.*, ACC for image classification) as well as the absolute relative error w.r.t. the file size condition. The performance metric is dependent on the utilized downstream task. Note for validation we quantize the predicted quantization tables of the control network and use the standard non-differentiable Pillow [92] JPEG implementation.

Image classification To validate the impact of our Deep Image Codec Control on the task of image classification we measure both the top-1 and top-5 accuracy. Since we target to preserve performance with reference to the prediction on the original (non-coded) image, we utilize the (hard) classification prediction obtained on the original image as a pseudo label for validation.

Semantic segmentation For the task of semantic segmentation, we follow standard practice and measure the mIoU (mean Intersection-over-Union) [55]–[58]. To obtain the mIoU, first, the per-class IoU is computed and averaged over all classes. We also report the pixel-wise ACC.

Object detection To measure the object detection performance we use the mAP (mean Average Precision) metric. We compute both the global mAP as well as the mAP for an IoU threshold of 50% (mAP_{50}) and 75% (mAP_{75}).

5.3. Baselines

5.3.1. Differentiable JPEG coding

We compare against the surrogate-based approaches by Xing *et al*. [34] and Shin *et al*. [33]. We also compare against the STE-based approach of Xie *et al*. [65]. Since Xie *et al*.are not modeling the JPEG quality, we extend the approach with the JPEG quality mapping of Xing *et al*. [34]. Both the approach of Shin *et al*.and Xie *et al*.offer no code, we have reimplemented both in PyTorch [96]. For the approach of Xing *et al*. [34], we use the official code. We noticed a bug in the official code (wrong quantization table 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.

5.3.2. Image Codec Control

As we are not aware of any existing baseline solving our constrained optimization problem (*c.f.* Eq. (4.8)), we compare our Deep Image Codec Control against standard JPEG. Standard JPEG, however, does not have an integrated file size control. To this end, we employ a binary search over the JPEG quality to adhere to the file size conditioning. In particular, we perform a binary search with the goal of finding a JPEG quality within -10% tolerance of the file size conditioning. Note this approach requires in the worst case $\lceil 99 \rceil$ encoding operations. If no matching JPEG quality can be found, the JPEG quality closest to the file size condition is used. As standard JPEG is optimized to minimize image distortion w.r.t. human quality assessment, we also utilize the optimized quantization tables for deep vision models, proposed by Luo *et al.* [47]. We will refer to the JPEG approach optimized for deep vision models as JPEG++.

5.4. Implementation Details

5.4.1. Differentiable JPEG coding

For all experiments, we utilize the OpenCV [46] JPEG implementation as a reference. We used JPEG coding with chroma downsampling (downsampling the chroma channels by a factor of 2). We utilize Kornia [109] for computing the SSIM. The utilized SSIM patch size is 11. Adversarial attack experiments are conducted using a ResNet-50 from torchvision [110] 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 $[0, 255]$.

5.4.2. Differentiable JPEG file size

For training our differentiable JPEG file size model we randomly sample quantization tables from a uniform distribution $U(1, 255)$ and regress the resulting file size. We utilize the Pillow [92] JPEG implementation since Pillow offers support for custom quantization tables. We pre-train a differentiable JPEG file size model for each downstream dataset. In particular, we perform 30k training step during pre-training and utilize a cosine learning rate schedule [111] (without annealing and restarts) with an initial learning rate of 10^{-4} and no weight-decay. For optimization we use the Adam optimizer [112]. To ensure a stable training we clip gradients with a norm larger than 1. Due to the different dataset resolution we utilize a batch size of 64 for ImageNet-1k, 32 for the COCO dataset, and 16 for Cityscapes. Note the differentiable JPEG file size models are fine tuned during end-to-end codec control training.

5.4.3. Deep Image Codec Control

We implement our Deep Image Codec Control using PyTorch [96], PyTorch Lightning [113], and Kornia [109]. For JPEG coding we utilize the Pillow [92] JPEG implementation as Pillow offers support for custom quantization tables. We utilize chroma downsampling in

JPEG coding. We train our control network for 20k steps using the Adam optimizer [112]. For the first 1.5k steps, we bootstrap the control network. The initial learning rate of the control network's conditional prediction head is set to 10^{-4}. For the ConvNeXt backbone, pre-trained on ImageNet-1k [53], we utilize an initial learning rate of 10^{-4}. The initial learning rate of the pre-trained JPEG file size model is set to 10^{-4}. We decay all learning rates using a cosine learning rate schedule without annealing and restarts [111]. We clip gradients with a norm larger than 0.1. For the task of semantic segmentation, we utilize a trained DeepLabV3 [26] model with ResNet-18 [52] backbone and a batch size of 28. For object detection, we use a trained DETR [8] model with ResNet-50 [52] backbone and a batch size of 64. Image classification is performed using a ResNet-152 [52] or ViT-S [14] with a batch size of 384. The loss weights are set to: $\lambda_{\mathrm{c}} = 30$, $\lambda_{\mathrm{p}} = 0.8$, $\lambda_{\mathrm{r}} = 0.2$, and $\lambda_{\mathrm{QT}} = 0.01$. To increase training efficiency, we utilize half-precision training. The target standard deviation $\zeta_{\boldsymbol{QT}}$ is set to 12. δ_{c} and δ_{r} are set to -0.02 and 0.02, respectively. Depending on the downstream task, we use different file size ranges. To determine the file size range we use the presented rate-distortion curves (*c.f.* Fig. 5.2). For the task of semantic segmentation (Cityscapes dataset), we use a file size range of 20kB to 100kB. For object detection (COCO dataset), we employ a file size range of 10kB to 100kB. For the downstream task of image classification, a range of 5kB to 50kB is used. We sample a random file size condition during training from a uniform distribution, using the aforementioned file size ranges. Since training can become unstable when a mini-batch does not capture the full range of file sizes, we chunk the uniform distribution into bins, corresponding to the used batch size. From each bin, a random file size is sampled. This ensures that each mini-batch does capture the full file size range, subsequently, stabilizing training with a low batch size. The differentiable file size model is trained in a lazy fashion every 12-th training step. Lazy training is used to ensure faster training, as each file size model training step requires JPEG encoding a mini-batch of images.

5.4.4. Computational setup

We train all models on a single GPU setup. For training the differentiable file size model we utilize a single NVIDIA A6000 GPU with 48GB of VRAM. For training our Deep Image Codec Control on the downstream task of semantic segmentation we follow the same computational setup. For the downstream tasks of image classification and object detection we use a single NVIDIA A100 GPU with 80GB of VRAM.

5.5. Results: Differentiable JPEG Coding

This section presents results of our differentiable JPEG coding approach. We report both the forward and backward performance. Further, we present thorough ablations of the proposed novel components.

5.5.1. Forward function results

We evaluate the ability to resemble the non-differentiable reference JPEG implementation of existing approaches against our differentiable JPEG approach (w/ STE). Our approach is able to resemble the non-differentiable JPEG implementation best. In particular, our approach outperforms all existing approaches over the whole JPEG quality range (*c.f.* Fig. 5.3). Especially, for strong compression strengths (low JPEG quality values) our proposed differentiable JPEG approach leads to better approximations. For weaker compression strengths (higher JPEG quality values) the performance gap between differentiable methods becomes smaller, but still, our approach leads to superior performance.

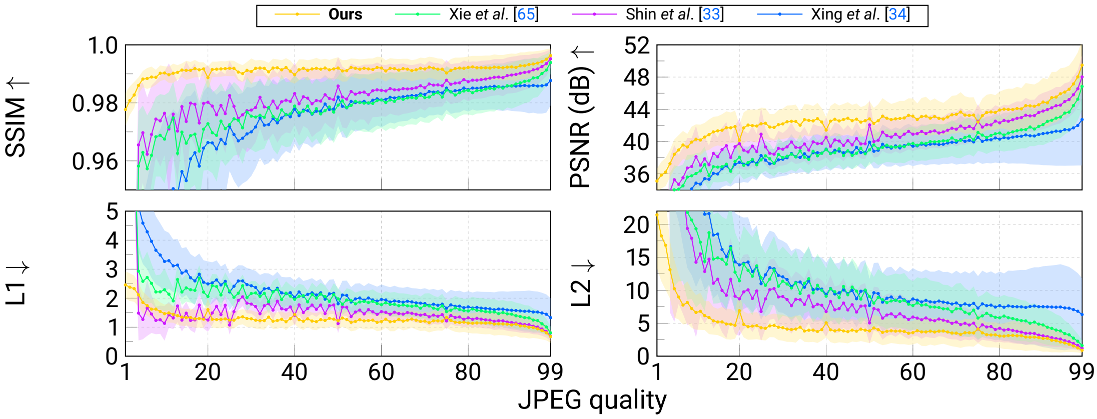


Figure 5.3. **Forward function performance.** Performance of approximating the reference JPEG implementation (OpenCV [46]) for different JPEG qualities. Mean and one standard deviation shown.

For small JPEG qualities (strong compression), we observe a significant disparity in performance between differentiable JPEG approaches. While our approach is still able to approximate standard non-differentiable JPEG well (SSIM > 0.97), all existing approaches fail and produce coded images vastly different from the reference implementation (SSIM < 0.97). These results are quantitatively analyzed in Fig. 5.4 and Tab. 5.1.

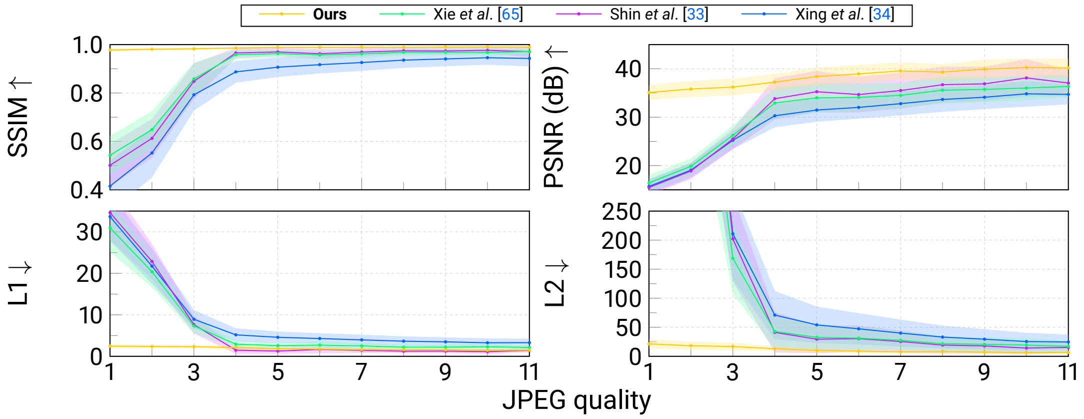


Figure 5.4. **Forward function performance for strong compression.** Performance of approximating the reference JPEG implementation (OpenCV [46]) for low JPEG qualities. Mean and one standard deviation shown.

	Configuration q range →	SSIM ↑ 1-99	SSIM ↑ 1-10	SSIM ↑ 11-99	PSNR ↑ 1-99	PSNR ↑ 1-10	PSNR ↑ 10-99
🟦	Xing *et al.* [34]	0.961	0.833	0.977	38.10	29.45	39.19
🟪	Xie *et al.* [65]	0.972	0.884	0.983	40.02	31.63	41.07
A	Shin *et al.* [33]	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 (*c.f.* Sec. 4.1.2) **(our differentiable STE JPEG)**	**0.993**	**0.993**	**0.992**	**43.49**	**41.14**	**43.78**

Table 5.1. **Forward function performance summary and ablation.** To ablate our approach, we gradually add our novel components to Shin *et al.* [33]. We also report the performance of other differentiable approaches. STE-based approaches marked in gray ▨.

Forward function ablation results. To understand what makes our differentiable JPEG implementation effective in resembling the reference implementation, we conduct an ablation study (*c.f.* Tab. 5.1 and Fig. 5.5). 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 and STE approaches. Using STE (config. *F*) further improves forward performance, leading to coded images perceptually indistinguishable from the reference implementation.

Fig. 5.5 showcases the effect of all introduced components for each (integer) JPEG quality. We observe that especially differentially clipping the coded output image (config. *E*) improves performance and, in particular, leads to strong results for small JPEG qualities. Interestingly, differentially clipping the quantization tables only leads to improvements for small JPEG quality values (strong compression). For larger JPEG quality values we observe no performance improvements by clipping the quantization tables. Clipping the output (coded) image in a differentiable manner leads to the valid pixel range leads to the most significant performance improvements over the whole JPEG quality range.

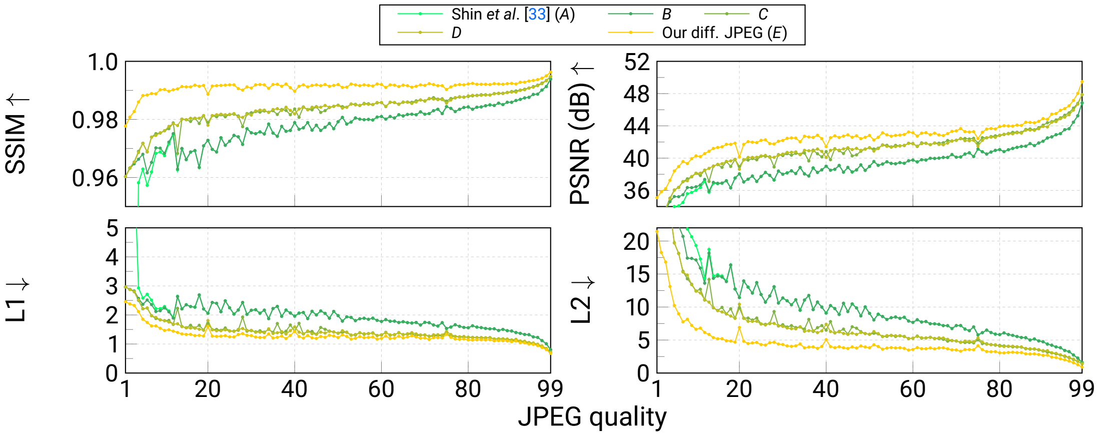


Figure 5.5. **Ablation study graph.** JPEG quality-wise forward function performance for different configurations (*c.f.* Tab. 5.1). Standard deviation is not shown for the sake of clarity.

Using STE (config. *F*) leads to a superior forward function performance over our non-STE approach (config. *E*, *c.f.* Fig. 5.5). As showcased in Fig. 5.6, our differentiable STE JPEG approach outperforms our differentiable JPEG approach w/o STE especially for low JPEG qualities. For high JPEG quality values (weak compression), our differentiable JPEG approach w/ and w/o STE lead to very similar results.

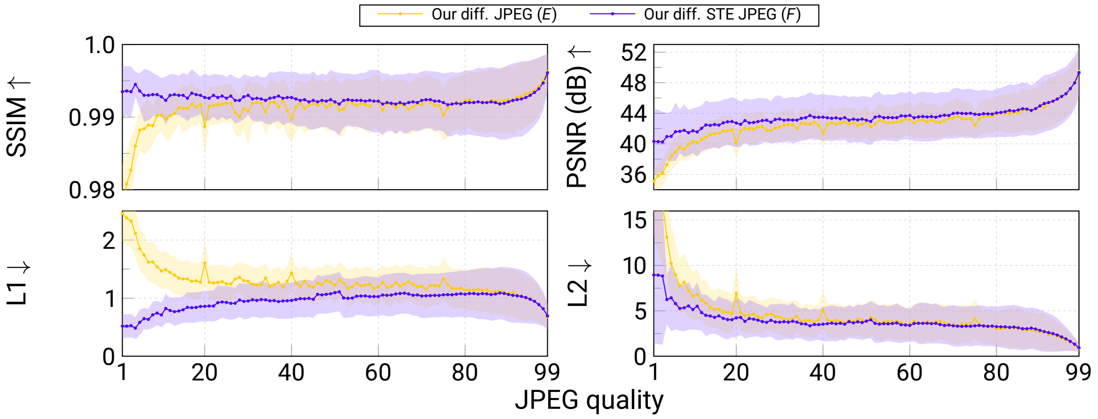


Figure 5.6. **Forward function performance with STE.** Performance of approximating the reference JPEG implementation (OpenCV [46]) for different JPEG qualties. Mean and one SD shown.

5.5.2. 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 (*c.f.* Tab. 5.2 (top)). When increasing ϵ to 9, our differentiable JPEG w/ STE scores slightly better than our approach w/o STE (*c.f.* Tab. 5.2 (bottom)).

When using IFGSM to craft adversarial examples, the results are consistently in favor of our differentiable JPEG approach w/o STE (*c.f.* Tab. 5.3). 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.

Our approach w/o STE leads to particularly strong attack results for low JPEG qualities compared to other approaches (*c.f.* Tab. 5.3). Our approaches (w/ and 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 (*c.f.* Fig. 5.7). In general, our approach w/ STE leads to a stronger forward performance, while a better backward performance is achieved w/o STE.

Approach	q range →	Top-1 acc ↓ 1-99	 1-10	 11-99	Top-5 acc ↓ 1-99	 1-10	 10-99
No attack		66.83	33.36	71.01	85.91	53.10	90.01
FSGM with $\epsilon = 3$							
Xing *et al.* [34]		53.92	26.00	57.42	77.90	45.65	81.93
Xie *et al.* [65]		41.48	20.07	44.15	66.53	38.52	70.03
Shin *et al.* [33]		*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.* [34]		51.25	27.64	54.20	75.24	47.41	78.72
Xie *et al.* [65]		38.29	18.15	40.81	61.89	35.38	65.21
Shin *et al.* [33]		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 5.2. **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.

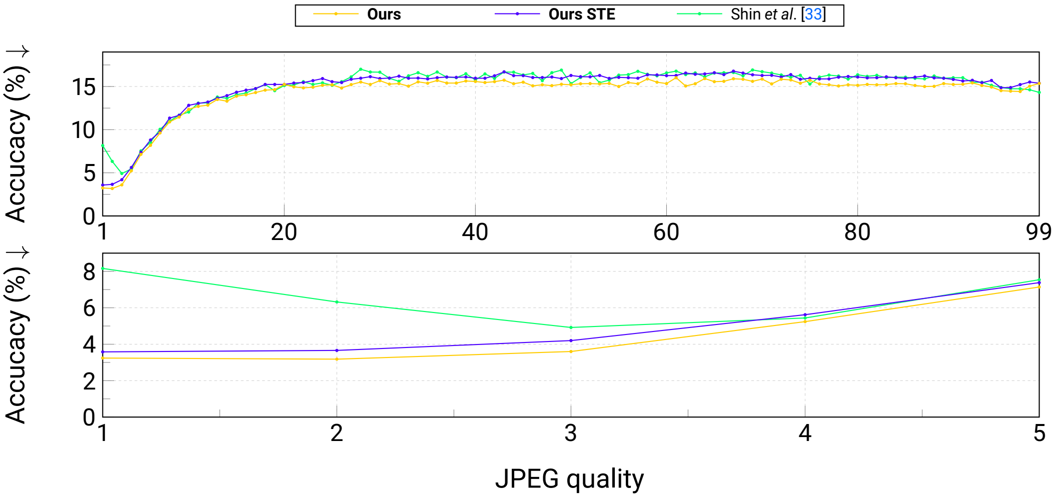


Figure 5.7. **IFGSM attack results.** Top-1 accuracy after IFGSM attack with our approach (w/ and w/o STE) *vs*. Shin *et al.* [33].

Approach	q range →	Top-1 acc ↓			Top-5 acc ↓		
		1-99	1-10	11-99	1-99	1-10	10-99
IFSGM with $\epsilon = 3$							
Xing *et al.* [34]		43.44	24.42	45.82	72.52	45.55	75.90
Xie *et al.* [65]		25.30	14.72	26.63	46.55	31.47	48.43
Shin *et al.* [33]		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.* [34]		39.59	24.99	41.41	67.73	45.41	70.52
Xie *et al.* [65]		15.03	8.70	15.82	27.34	19.21	28.35
Shin *et al.* [33]		*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 5.3. **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 $\epsilon = 9$.

Vanishing gradient experiment. Both our differentiable JPEG w/ and w/o STE lead to lower gradient norms at the desired minima (*c.f.* Tab. 5.4). 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.

Approach	q range →	$\mathbb{E}_{Q,\mathcal{D}}[\|\|\Delta_q \mathcal{L}_1\|\|]$			$\mathbb{E}_{Q,\mathcal{D}}[\|\|\Delta_{\boldsymbol{QT}_{Y,C}} \mathcal{L}_1\|\|]$		
		1-99	1-10	11-99	1-99	1-10	10-99
Xing *et al.* [34]		1.254	7.271	0.502	–	–	–
Xie *et al.* [65]		0.955	4.617	0.492	0.913	1.041	0.896
Shin *et al.* [33]		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 5.4. **Vanishing gradient results.** Average gradient norms at desired global minima. Gradient norms are averaged over the integer JPEG quality range Q and the Set14 [50] dataset $\mathcal{D}$.

5.5.3. Additional ablation results

Which rounding/floor approximation to use? In Tab. 5.5, 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 $\frac{1}{2}\tanh(5\,(x - \frac{1}{2} - \lfloor x \rceil)) + \frac{1}{2}$		0.972	0.918	0.978	38.51	31.15	39.42

Table 5.5. **Rounding/floor ablation (forward function).** Performance of approximating the reference OpenCV [46] 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 (*c.f.* Tab. 5.6). This suggests gradients derived from the polynomial approximation are more "useful" than from other approximations. While leading to a fair forward performance (*c.f.* Tab. 5.5), the Fourier approximation leads to poor attack results, thus also yields worse gradients. This result aligns with the attack results by Xing *et al.* (*c.f.* Tab. 5.2 and Tab. 5.3) 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 5.6. **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 (*c.f.* Tab. 5.5 and Tab. 5.6), 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 (*c.f.* Tab. 5.7), 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 5.7. **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.

5.6. Results: Differentiable JPEG File Size Modeling

Here we validate the performance of our differentiable JPEG file size model. We generate quantization tables by our trained Deep Image Codec Control for 10 different file size conditions, if not stated differently. We feed the quantization tables and the current image into our differentiable JPEG file size model and regress the JPEG file size. We measure the MARE or error between the prediction of our differentiable JPEG file size model and the JPEG file size generated by standard non-differentiable JPEG.

5.6.1. Results on Cityscapes

Fig. 5.8 presents differentiable file size model results on the Cityscapes dataset [55]. For a low target file size, our differentiable JPEG file size model leads to a MARE of about 0.025. In case of higher target file sizes the performance of our differentiable JPEG file size model is slightly reduced. However, the error does not exceed a MARE of 0.05 for large target file sizes. In general, the differentiable JPEG file size model performance on Cityscapes leads to accurate differentiable file size estimates over the full file size range, facilitating the learning of our Deep Image Codec Control.

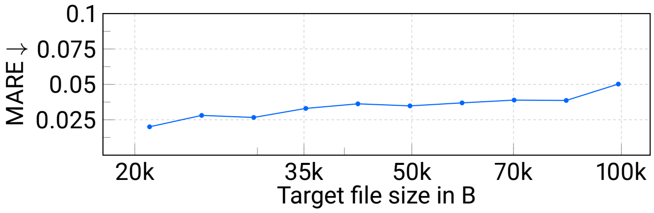


Figure 5.8. **Differentiable JPEG file size model results (Cityscapes).** MARE of our differentiable JPEF file size model on Cityscapes (in blue 🟦) using our Deep Image Codec Control to predict quantization tables for the respective file size conditions.

5.6.2. Results on COCO

On the COCO dataset, we observe a similar performance as on the Cityscapes dataset. Fig. 5.9 visualizes the obtained results on COCO. As on Cityscapes, our differentiable JPEG file size model on COCO leads to stronger results for low target file sizes and slightly deteriorates in performance for higher target file sizes. Still, our differentiable JPEG file size model only leads to a MARE of about 0.06 in the worst case.

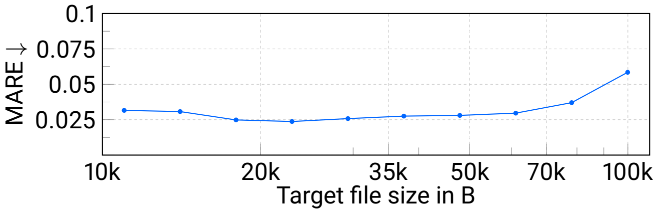


Figure 5.9. **Differentiable JPEG file size model results (COCO).** MARE of our differentiable JPEF file size model on COCO (in blue 🟦) using our Deep Image Codec Control to predict quantization tables for the respective file size conditions.

5.6.3. Results on ImageNet

Fig. 5.10 presents differentiable file size results obtained on ImageNet. We observe a similar trend as on Cityscapes and COCO that our differentiable file size model performs better for small target file sizes. However, the performance deterioration for larger target file sizes is more severe than on Cityscapes and ImageNet. This deterioration in performance might be explained by the variability of the ImageNet dataset compared to Cityscapes. As, in particular, for larger target file sizes the variance of the generated file size increases (*c.f.* Fig. 3.1) our differentiable JPEF file size model might suffer to generate a more accurate file size predictions on ImageNet. While we train for the same number of iterations on all datasets, scaling up the ImageNet training might lead to more accurate differentiable file size results.

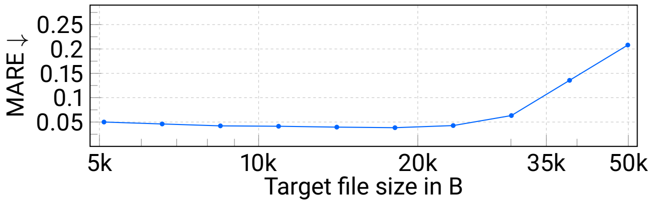


Figure 5.10. **Differentiable JPEG file size model results (ImageNet).** MARE of our differentiable JPEF file size model on ImageNet (in blue 🟦) using our Deep Image Codec Control to predict quantization tables for the respective file size conditions.

5.6.4. Error distribution

In Fig. 5.11, we report the error distribution of our differentiable JPEG file size model. We visualize the error histogram of our differentiable JPEG file size model using the Cityscapes dataset and randomly generated quantization tables. The error distribution entails a symmetric shape, this is beneficial for our Deep Image Codec Control as the differentiable file size model does not introduce a significant systematic error w.r.t. to JPEG file size.

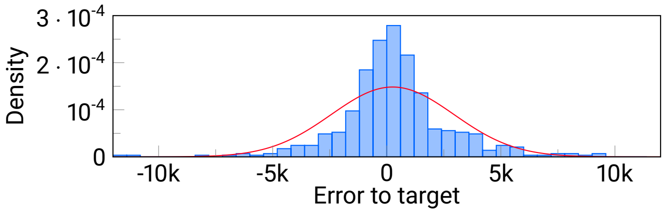


Figure 5.11. **Differentiable file size model error distribution.** We report the error distribution (in blue 🟦) of our differentiable file size model on Cityscapes [55]. We sample random quantization tables and measure the error to the true JPEG file size. Gaussian fit of the error distribution plotted in red 🟥.

5.7. Results: Deep Image Codec Control

This section provides validation results of our Deep Image Codec Control for three different computer vision tasks: semantic segmentation, object detection, and image classification. We also provide results when transferring a Deep Image Codec Control to a different downstream deep vision model. Finally, we also provide insides of failed experiments.

5.7.1. Semantic segmentation results

In Fig. 5.12 we show results of our Deep Image Codec Control in preserving the semantic segmentation downstream performance of a DeepLabV3 network with ResNet-50 backbone. Our Deep Image Codec Control is able to adapt to different target file sizes. When measuring the pixel-wise ACC our Deep Image Codec Control better preserves semantic segmentation performance for low target file sizes (*c.f.* Fig. 5.12 right). When measuring the mIoU our Deep Image Codec Control falls short (*c.f.* Fig. 5.12 left). We suspect that our control network might cheat and predict quantization tables optimizing for more present classes. As the mIoU metric weights pixels non-uniformly, a shortcut prediction by the control network might explain the disparity between both metrics.

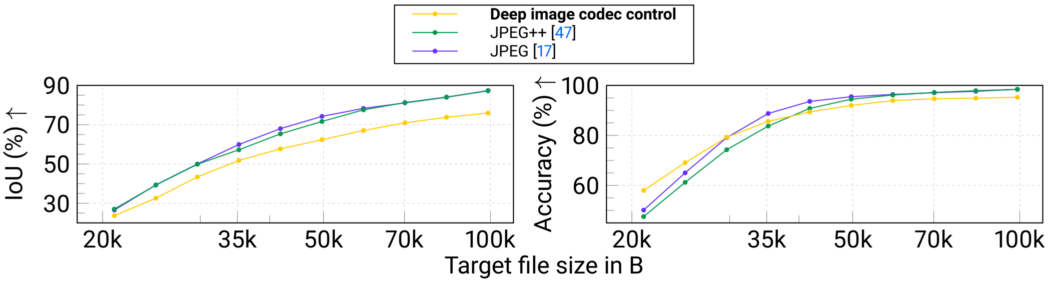


Figure 5.12. **Deep Image Codec Control semantic segmentation results.** Validation results of our Deep Image Codec Control and baselines for different target file sizes. A DeepLabV3 model w/ ResNet-18 backbone is utilized as the downstream deep vision model.

In Fig. 5.13, we report the ability to meet a dynamic target file size. Our Deep Image Codec Control is able to predict quantization tables leading to a file size close to the target file size. For small target file sizes, we outperform both baselines. For larger target file sizes (above 50kB) our Deep Image Codec Control performs on par with the baselines. We suspect that this behavior is induced by our differentiable JPEG file size model. As our differentiable JPEG file size model performs slightly weaker for larger target file sizes this error could potentially also have a negative impact on the Deep Image Codec Control performance for the respective target file size range.

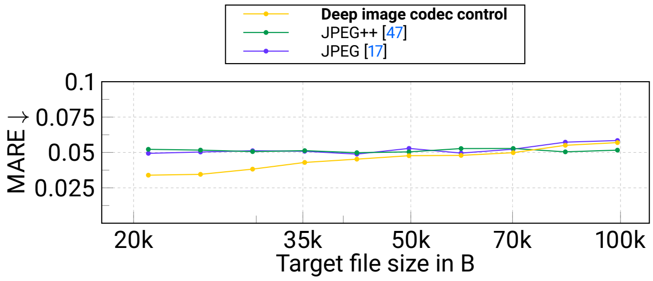


Figure 5.13. **Deep Image Codec Control file size condition results (semantic segmentation).** Validation results of our Deep Image Codec Control and baselines of meeting a dynamic file size condition. We report the MARE of the actual file size to the target file size.

5.7.2. Object detection results

For the task of object detection, we showcase results in Fig. 5.14. Similarly, to the task of semantic segmentation, our Deep Image Codec Control adapts to different target file sizes, however, falls short of the baselines in terms of preserving downstream performance.

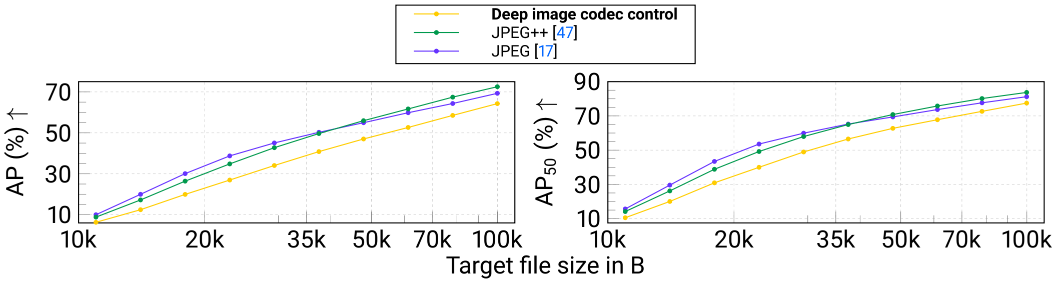


Figure 5.14. **Deep Image Codec Control object classification results.** Validation results of our Deep Image Codec Control and baselines for different target file sizes. A DETR model w/ ResNet-50 backbone is utilized as the downstream deep vision model.

While falling short in preserving downstream performance, our Deep Image Codec Control follows accurately a dynamic file size condition. We provide file size condition results for the task of object detection on COCO in Fig. 5.15. Our Deep Image Codec Control leads to a file size closer to the target file size as both baselines. Similar to the results on semantic segmentation our Deep Image Codec Control tends to more accurately follow the target file size for low file sizes.

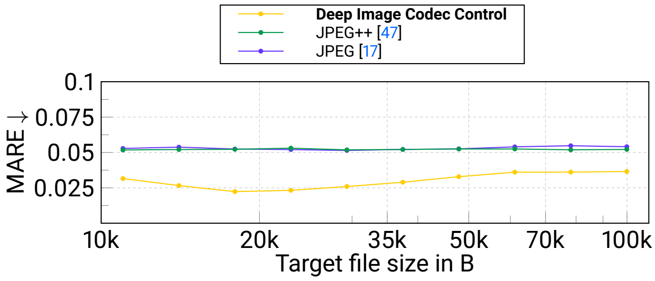


Figure 5.15. **Deep Image Codec Control file size condition results (object detection).** Validation results of our Deep Image Codec Control and baselines of meeting a dynamic file size condition. We report the MARE of the actual file size to the target file size.

5.7.3. Image classification results

For the task of image classification, we report results for two different downstream models, ViT-S and ResNet-152. The downstream results for different file size conditions are reported in Fig. 5.16 and Fig. 5.17, respectively. While we meet the performance of our JPEG baseline for a file size of 12kB, we especially fall short for larger file size conditions (> 12kB). While we bootstrap our control network on the quantization tables predicted by JPEG++ our Deep Image Codec Control diverges from this solution. This might be caused by the strong quantization table regularization we impose.

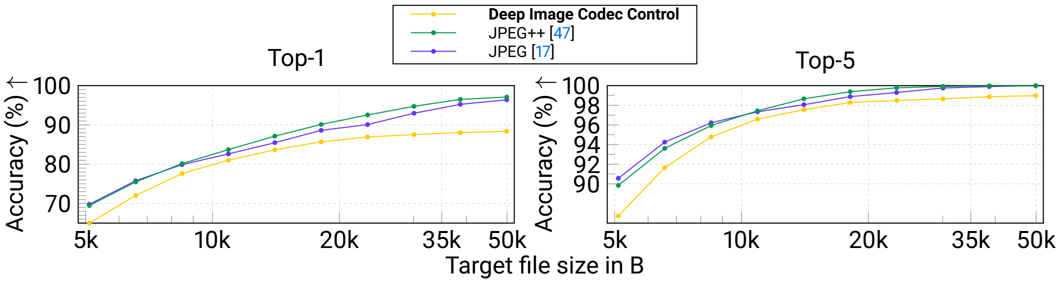


Figure 5.16. **Deep Image Codec Control object classification results.** Validation results of our Deep Image Codec Control and baselines for different target file sizes. A ViT-S image classification model is utilized as the downstream deep vision model.

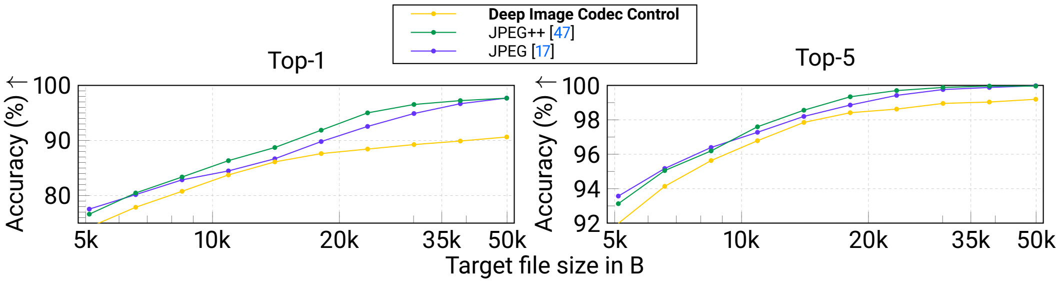


Figure 5.17. **Deep Image Codec Control object classification results.** Validation results of our Deep Image Codec Control and baselines for different target file sizes. A ResNet-152 image classification model is utilized as the downstream deep vision model.

In Fig. 5.18 and Fig. 5.19 we visualize the dynamic file size condition results for the respective downstream model (ViT-S and ResNet-15). For a target file size below 30kB we achieve a strong performance and accurately meet the utilized target file size. For larger file size conditions our Deep Image Codec Control struggles to target the target file size. This is potentially caused by the error introduced by our differentiable JPEG file size model (*c.f.* Fig. 5.10). Interestingly, both baselines also struggle to closely approach the correct target file size. This is potently caused by the fact that the JPEG quality offers a too coarse grain control for high file size conditions (*c.f.* Fig. 5.2).

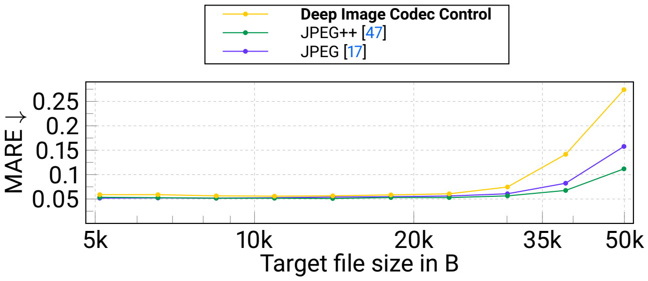


Figure 5.18. **Deep Image Codec Control file size condition results (image classification w/ ViT-S).** Validation results of our Deep Image Codec Control and baselines of meeting a dynamic file size condition. We report the MARE of the actual file size to the target file size.

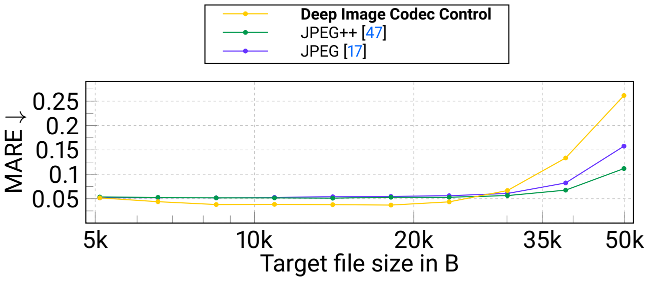


Figure 5.19. **Deep Image Codec Control file size condition results (image classification w/ ResNet-152).** Validation results of our Deep Image Codec Control and baselines of meeting a dynamic file size condition. We report the MARE of the actual file size to the target file size.

5.7.4. Transfer results (image classification)

We also tested the transferability of Deep Image Codec Controls between different downstream models. In particular, we tested a Deep Image Codec Control, trained using a ViT-S downstream model, to preserve the performance of a ResNet-152 image classification model. These results are depicted in Fig. 5.20. Compared to the Deep Image Codec Control directly trained with a ResNet-152 model performance of the transferred Deep Image Codec Control is reduced by about 1.5% in top-1 ACC. Interestingly, this performance gap becomes narrower for target file sizes above 20kB. These findings align with the work of Chen *et al.* [31]. They show when using a deep video codec, trained utilizing a ResNet-50, to preserve the performance of a VGG-19 network [114] downstream deteriorates over a deep video codec directly trained on the respective downstream model.

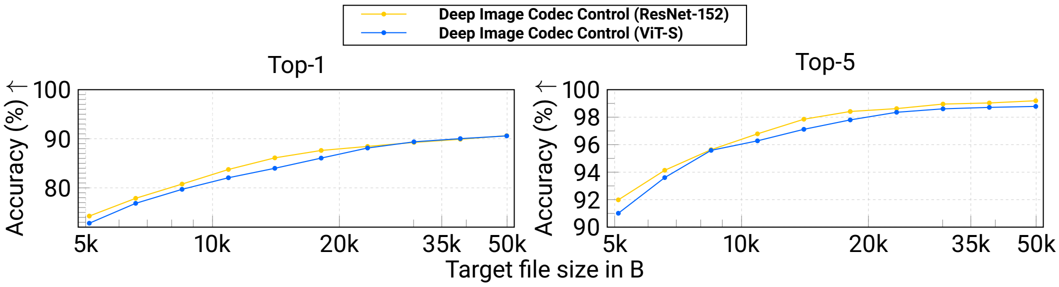


Figure 5.20. **Deep Image Codec Control downstream model transfer results.** We transfer a Deep Image Codec Control trained using a ViT-S model to preserve the downstream performance of a different downstream model (ResNet-152) during inference. For reference, we also visualize the performance of a Deep Image Codec Control trained on a ResNet-152 model.

5.7.5. Failed Deep Image Codec Control experiments

This subsection provides results when quantization table regularization ($\mathcal{L}_{QT}$) is not considered during training our Deep Image Codec Control. Bootstrap training is also omitted. We observe that training becomes unmanageably unstable. This behavior is showcased in Fig. 5.21 in which we visualize the control networks loss curve ($\mathcal{L}_{\mathrm{c}}$ w/o $\mathcal{L}_{QT}$). While the loss before 10k tends to recover after severe spikes training becomes more and more unstable as the training progresses. For reference, when considering our quantization table regularizer the training loss converges below a value of 0.2.

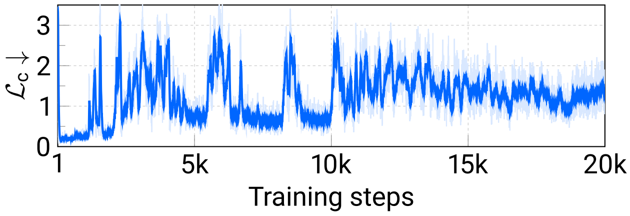


Figure 5.21. **Training loss w/o quantization table regularization for the downstream task of semantic segmentation.** Visualization of the control networks training loss $\mathcal{L}_{\mathrm{c}}$ for 20k training iterations. For visualization purposes, we show an exponential moving average of the loss overplayed with the raw loss values. No bootstrap training phase was used.

When visualizing only the (unscaled) performance loss we observe an even more unstable behavior w.r.t. the downstream performance (*c.f.* Fig. 5.22). This demonstrates that the control network is not able to preserve downstream performance in the unregularized setting (not considering $\mathcal{L}_{QT}$).

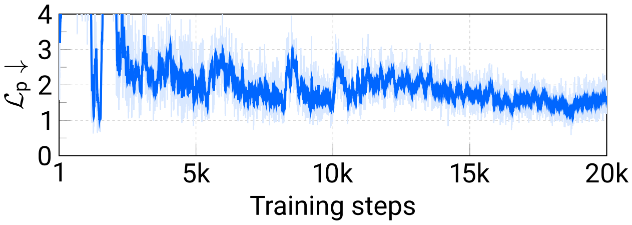


Figure 5.22. **Training performance loss w/o quantization table regularization for the downstream task of semantic segmentation.** Visualization of the control networks training performance loss $\mathcal{L}_{\mathrm{p}}$ for 20k training steps. For visualization purposes, we show an exponential moving average of the loss overplayed with the raw loss values. No bootstrap training phase was used.

We also performed preliminary experiments by replacing the proposed quantization table regularizer $\mathcal{L}_{QT}$ with a smoothness regularizer. In particular, we experimented with regularizing the first or second-order image gradient of the quantization tables, imposing the control network to predict similar quantitation values for neighboring frequency components. Note this regularization is similar to the smoothness terms used in unsupervised optical flow prediction [115]. However, both first and second-order smoothness could not help stabilize the training process.

6. Discussion & Future Work

This chapter discusses the presented methodology. We also present possible approaches to overcome existing limitations. Finally, we also pose and discuss additional research questions beyond the scope of this thesis which might be relevant for future research.

6.1. Discussion

Training a content and file size adaptive Deep Image Codec Control is feasible, however, requires strong regularization. The control network is required to both explore possible codec parameters w.r.t. a dynamic file size condition and downstream deep vision performance. This is, in particular, challenging as multiple codec parameters might optimize our control objective (*c.f.* Eq. (4.8)). Optimizing over a complex loss surface of the downstream deep vision model introduces additional complexity. When using no regularization training is highly unstable and does not converge to a strong solution (*c.f.* Sec. 5.7.5). However, regularizing the prediction of the control network constrains the space of possible solutions. This potentially leads to suboptimal predictions by the control network. Thus, finding the correct regularization strategy is significant for learning a strong Deep Image Codec Control and preserving downstream vision performance. A good regularization strategy is required to stabilize the end-to-end training while not omitting strong solutions from the solution space. We argue that while stabilizing the training our proposed regularization might offer a too strong constraint for significantly outperforming existing approaches. To this end, we suggest possible techniques to overcome the current limitations of our Deep Image Codec Control in the following section.

6.2. Overcoming Existing Limitations

Here we present possible approaches to overcome the current limitations of our proposed Deep Image Codec Control.

Curriculum learning. Choi *et al.* [63] demonstrated that learning the dynamic prediction of quantization tables for a fixed target file size outperforms existing approaches. Starting training with a single target file size and progressively enlarging the range of file sizes might offer a more stable training process. This curriculum learning [116]–[118] approach would constrain the control network to first learn codec parameters relevant to preserve downstream performance and to meet a single target file size. By progressively enlarging the range of file sizes the control network can transfer the learned knowledge to a wide range of dynamic file size conditions. By first constraining the solution space and progressively enlarging the solution space this approach might help stabilizing our end-to-end codec control training. Note this approach can also be seen as a kind of regularization strategy [116].

Attribution map regularization. As the gradient-based feedback from the downstream deep vision model is potentially very sparse utilizing a regularizer offering a more dense gradient feedback w.r.t. downstream performance might improve learning [20], [99]. In particular, enforcing a similarity between an attribution map obtained on the original image and an attribution map from the coded image might offer a potentially interesting approach. By regularizing the attribution maps the control network would be forced to preserve relevant features, subsequently preserving downstream performance. Additionally, the gradient signal from this regularizer might be potentially more dense than just gradient-based feedback from the downstream model. This regularization approach would also not impose a vast constraint on the quantization table predictions. As attribution maps often require multiple backward passes through a downstream model developing a performant and general regularization approach might pose significant challenges [22]. Lazy regularization [119] might pose a potential avenue for efficiently regularizing the control network using attribution maps.

Quantization table decomposition. Quantization tables often entail regularities [17], [47], [80]. Predicting decomposed quantization tables, instead of a full rank matrix might ease learning. In particular, predicting two vectors per quantization table and building the full quantization table by the outer product could potentially stabilize training. The control network, thus, only needs to predict 16 quantization values instead of 8×8. While this approach constrains the space of possible quantization tables it might impose a bias usefully for stabilizing and enhancing learning.

6.3. Future Work

This thesis focuses on showcasing the feasibility of training one Deep Image Codec Control for one specific downstream task and model. However, training a control for each downstream task and model is not practical. To facilitate the widespread applicability of a Deep Image Codec Control future work might consider the support for multiple downstream models. Generalizing over multiple tasks might also pose an interesting direction for future research. More specifically, prompt tuning the control network for multiple downstream tasks as done by Chen *et al*. [31] might be an interesting avenue to explore. As our control network explores the importance of features in frequency space connecting this work with recent findings on how CNNs and ViTs perceive images might offer potentially useful insides [120].

7. Conclusion

Motivated by the fact that current deep vision models base their downstream predictions on silent parts of frequencies of an image, this thesis presented a novel approach for optimizing JPEG for deep vision models [19]–[22], [25]. As JPEG is developed to minimize image distortion w.r.t. human quantity assessment and not w.r.t. deep vision performance this thesis analyzes the impact of JPEG coding on downstream deep vision performance. We performed experiments using JPEG-coded images on three common computer vision tasks: image classification, object detection, and semantic segmentation and a variety of different model architectures (*e.g.*, ResNets, and ViTs). All models suffer from a vast deterioration in downstream performance when utilizing JPEG-coded images during inference. For strong compression, downstream deep vision performance almost completely collapses.

To approach the vast deterioration in downstream performance, we present an approach to augment/control JPEG w.r.t. downstream deep vision performance. In particular, we present a novel Deep Image Codec Control for JPEG. Our Deep Image Codec Control aims to control JPEG such that downstream deep vision performance is preserved and a dynamic file size condition is met. We also adhere to existing standardizations to facilitate the possible widespread applicability of our approach.

As standard JPEG encoding-decoding is non-differentiable, the direct application of end-to-end gradient-based learning is not feasible. To overcome this non-differentiability, this thesis presents a novel differentiable JPEG approach. The proposed differentiable JPEG approach supports gradient w.r.t. the input image, the JPEG quality, the quantization tables, and the color conversion parameters. In contrast to the existing differentiable JPEG approaches our approach is the first to accurately approximate JPEG over the full compression range while providing gradient approximations effective for gradient-based optimization.

While our differentiable JPEG coding approach allows the properation of gradient-based feedback through the JPEG encoding-decoding, the file size of the JPEG encoded image is not modeled. To this end, we also present a novel differentiable JPEG file size surrogate model. This enables the propagation of gradient-based feedback w.r.t. the JPEG file size. Given an input image and JPEG quantization tables, our differentiable JPEG file size model is able to accurately regress the resulting JPEG file size and provide effective gradient-based feedback.

The proposed differentiable JPEG encoding-decoding approach paired with our differentiable JPEG file size surrogate model enables us to formulate the learning of our Deep Image Codec Control as an end-to-end learning problem. In particular, we present a novel end-to-end self-supervised training formulation to learn our Deep Image Codec Control. By utilizing pseudo labeling based on prediction obtained on the original (non-coded) images our approach circumvents the use of human annotations. Our experiments on three common computer vision tasks showcase that our Deep Image Codec Control is able to adapt to different file size conditions during inference and can preserve downstream performance. While mostly reaching an on-par performance to existing approaches, our end-to-end learnable Deep Image Codec Control offers a novel and alternative approach for optimizing standard image codecs for machine vision. We thoroughly discuss the current limitations and difficulties of our Deep Image Codec Control and suggest multiple potential improvements to be explored by future research.

Acknowledgements

I acknowledge the support of NEC Laboratories America, Inc. and my supervisors Biplob Debnath, Oliver Hahn, and Stefan Roth. I also thank my other supervisors and colleagues at NEC Labs: Deep Patel, Srimat Chakradhar, and Oliver Po for their amazing support throughout this thesis and other projects.

A special thanks to *Tim Prangemeier* for the amazing support throughout my bachelor thesis and numerous other research projects! I would also like to thank my previous supervisors, co-authors, and supporters: André Oliveira Françani, Christian Wildner, Christoph Hoog Antink, Maurice Rohr, Ozdemir Cetin, and Tim Kircher (in alphabetic order). I also acknowledge the support from Italy by Marius Memmel. Finally, I would like to thank Markus Baier for his aid with the computational setup and late night fixing my server problem.

Bibliography

[1] C. Reich, B. Debnath, D. Patel, and S. Chakradhar, “Differentiable JPEG: The Devil is in the Details”, in *WACV (to appear)*, 2024.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press.

[3] C. Reich, “Generation and Simulation of Yeast Microscopy Imagery with Deep Learning”, *arXiv:2103.11834*, 2021.

[4] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”, *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[5] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, in *ICML*, 2015, pp. 448–456.

[7] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, “Modulating early visual processing by language”, in *NeurIPS*, vol. 30, 2017.

[8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers”, in *ECCV*, 2020, pp. 213–229.

[9] I. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples”, in *ICLR*, 2015.

[10] Y. Wu and K. He, “Group Normalization”, in *ECCV*, 2018, pp. 3–19.

[11] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial Machine Learning at Scale”, in *ICLR*, 2017.

[12] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world”, in *Artificial Intelligence Safety and Security*, 2018, pp. 99–112.

[13] Y. Bengio, N. Léonard, and A. Courville, “Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation”, *arXiv:1308.3432*, 2013.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, in *ICLR*, 2020.

[15] F. K. Došilović, M. Brčić, and N. Hlupić, “Explainable artificial intelligence: A survey”, in *MIPRO*, 2018, pp. 0210–0215.

[16] A. Ali, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, I. Laptev, N. Neverova, G. Synnaeve, J. Verbeek, and H. Jegou, “XCiT: Cross-Covariance Image Transformers”, in *NeurIPS*, 2021, pp. 20 014–20 027.

[17] G. K. Wallace, “The JPEG still picture compression standard”, *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[18] G. Hudson, A. Léger, B. Niss, I. Sebestyén, and J. Vaaben, “JPEG-1 standard 25 years: past, present, and future reasons for a success”, *Journal of Electronic Imaging*, vol. 27, no. 4, pp. 040901-1–040901-19, 2018.

[19] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic Attribution for Deep Networks”, in *ICML*, 2017, pp. 3319–3328.

[20] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”, in *ICCV*, 2017, pp. 618–626.

[21] R. Shetty, B. Schiele, and M. Fritz, “Not Using the Car to See the Sidewalk — Quantifying and Controlling the Effects of Context in Classification and Segmentation”, in *CVPR*, 2019, pp. 8218–8226.

[22] R. Hesse, S. Schaub-Meyer, and S. Roth, “Fast Axiomatic Attribution for Neural Networks”, in *NeruIPS*, vol. 34, 2021, pp. 19 513–19 524.

[23] A. Otani, R. Hashiguchi, K. Omi, N. Fukushima, and T. Tamaki, “Performance Evaluation of Action Recognition Models on Low Quality Videos”, *IEEE Access*, vol. 10, pp. 94 898–94 907, 2022.

[24] C. Reich, B. Debnath, D. Patel, T. Prangemeier, and S. Chakradhar, “Deep Video Codec Control”, *arXiv:2308.16215*, 2023.

[25] S. Wang, R. Veldhuis, C. Brune, and N. Strisciuglio, “What do neural networks learn in image classification? A frequency shortcut perspective”, in *ICCV*, 2023, pp. 1433–1442.

[26] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking Atrous Convolution for Semantic Image Segmentation”, *arXiv:1706.05587*, 2017.

[27] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy Image Compression with Compressive Autoencoders”, in *ICLR*, 2016.

[28] D. Minnen, J. Ballé, and G. D. Toderici, “Joint autoregressive and hierarchical priors for learned image compression”, in *NeurIPS*, vol. 31, 2018.

[29] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, “Deep Learning-Based Video Coding: A Review and a Case Study”, *ACM Computing Surveys*, vol. 53, no. 1, pp. 1–35, 2020.

[30] M. Song, J. Choi, and B. Han, “Variable-Rate Deep Image Compression through Spatially-Adaptive Feature Transform”, in *ICCV*, 2021, pp. 2380–2389.

[31] Y.-H. Chen, Y.-C. Weng, C.-H. Kao, C. Chien, W.-C. Chiu, and W.-H. Peng, “TransTIC: Transferring Transformer-based Image Compression from Human Perception to Machine Perception”, in *ICCV*, 2023, pp. 23 297–23 307.

[32] J. Ascenso, E. Alshina, and T. Ebrahimi, “The JPEG AI Standard: Providing Efficient Human and Machine Visual Data Consumption”, *IEEE Multimedia*, vol. 30, no. 1, pp. 100–111, 2023.

[33] R. Shin and D. Song, “JPEG-resistant Adversarial Images”, in *NIPS Workshop on Machine Learning and Computer Security*, vol. 1, 2017, p. 8.

[34] Y. Xing, Z. Qian, and Q. Chen, “Invertible Image Signal Processing”, in *CVPR*, 2021, pp. 6287–6296.

[35] R. Hataya, J. Zdenek, K. Yoshizoe, and H. Nakayama, “Faster AutoAugment: Learning Augmentation Strategies using Backpropagation”, in *ECCV*, 2020, pp. 1–16.

[36] A. B. Jung, K. Wada, J. Crall, S. Tanaka, *et al.*, *imgaug*, https://github.com/aleju/imgaug, 2020.

[37] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training Generative Adversarial Networks with Limited Data”, in *NeurIPS*, vol. 33, 2020, pp. 12 104–12 114.

[38] J. Shi, E. Riba, D. Mishkin, F. Moreno, and A. Nicolaou, “Differentiable Data Augmentation with Kornia”, *arXiv:2011.09832*, 2020.

[39] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning”, *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[40] M. Shu, Y. Shen, M. C. Lin, and T. Goldstein, “Adversarial Differentiable Data Augmentation for Autonomous Systems”, in *ICRA*, 2021, pp. 14 069–14 075.

[41] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han, “Differentiable Augmentation for Data-Efficient GAN Training”, in *NeurIPS*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, 2020, pp. 7559–7570.

[42] C. Zhang, A. Karjauv, P. Benz, and I. S. Kweon, “Towards Robust Data Hiding Against (JPEG) Compression: A Pseudo-Differentiable Deep Learning Approach”, *arXiv:2101.00973*, 2020.

[43] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, “HiDDeN: Hiding Data With Deep Networks”, in *ECCV*, 2018, pp. 657–672.

[44] Y. Yang, C. Liang, H. He, X. Cao, and N. Z. Gong, “FaceGuard: Proactive Deepfake Detection”, *arXiv:2109.05673*, 2021.

[45] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, “Countering Adversarial Images using Input Transformations”, in *ICLR*, 2018.

[46] G. Bradski, “The OpenCV Library”, *Dr. Dobb’s Journal*, vol. 25, no. 11, pp. 120–123, 2000.

[47] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar, “The Rate-Distortion-Accuracy Tradeoff: JPEG Case Study”, in *DCC*, 2021, pp. 354–354.

[48] A. J. Hussain, A. Al-Fayadh, and N. Radi, “Image Compression Techniques: A Survey in Lossless and Lossy algorithms”, *Neurocomputing*, vol. 300, pp. 44–69, 2018.

[49] D. L. MacAdam, “Visual Sensitivities to Color Differences in Daylight”, *Journal of the Optical Society of America*, vol. 32, no. 5, pp. 247–274, 1942.

[50] R. Zeyde, M. Elad, and M. Protter, “On Single Image Scale-Up Using Sparse-Representations”, in *Curves and Surfaces: 7th International Conference*, 2012, pp. 711–730.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge”, *IJCV*, vol. 115, pp. 211–252, 2015.

[52] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, in *CVPR*, 2016, pp. 770–778.

[53] R. Wightman, *PyTorch Image Models*, https://github.com/rwightman/pytorch-image-models, 2019.

[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context”, in *ECCV*, 2014, pp. 740–755.

[55] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding”, in *CVPR*, 2016, pp. 3213–3223.

[56] MMSegmentation Contributors, *MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark*, `https://github.com/open-mmlab/mmsegmentation`, 2020.

[57] N. Araslanov and S. Roth, "Self-supervised Augmentation Consistency for Adapting Semantic Segmentation", in *CVPR*, 2021, pp. 15 384–15 394.

[58] S. Bahmani, O. Hahn, E. Zamfir, N. Araslanov, D. Cremers, and S. Roth, "Semantic Self-adaptation: Enhancing Generalization with a Single Sample", *TMLR*, 2023, ISSN: 2835-8856.

[59] P. Esser, R. Rombach, and B. Ommer, "Taming Transformers for High-Resolution Image Synthesis", in *CVPR*, 2021, pp. 12 873–12 883.

[60] E. Jang, S. Gu, and B. Poole, "Categorical Reparameterization with Gumbel-Softmax", in *ICLR*, 2017.

[61] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing Flows for Probabilistic Modeling and Inference", *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 2617–2680, 2021.

[62] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning", in *NIPS*, vol. 30, 2017.

[63] J. Choi and B. Han, "Task-Aware Quantization Network for JPEG Image Compression", in *ECCV*, 2020, pp. 309–324.

[64] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy Image Compression with Compressive Autoencoders", in *ICLR*, 2017.

[65] X. Xie, N. Zhou, W. Zhu, and J. Liu, "Bandwidth-Aware Adaptive Codec for DNN Inference Offloading in IoT", in *ECCV*, 2022, pp. 88–104.

[66] Y. Strümpler, R. Yang, and R. Timofte, "Learning to Improve Image Compression without Changing the Standard Decoder", in *ECCVW*, 2020, pp. 200–216.

[67] Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, "Differentiable Automatic Data Augmentation", in *ECCV*, 2020, pp. 580–595.

[68] Z. Chen, K. Fan, S. Wang, L.-Y. Duan, W. Lin, and A. Kot, "Lossy Intermediate Deep Learning Feature Compression and Evaluation", in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2414–2422.

[69] S. Singh, S. Abu-El-Haija, N. Johnston, J. Ballé, A. Shrivastava, and G. Toderici, "End-to-End Learning of Compressible Features", in *ICIP*, IEEE, 2020, pp. 3349–3353.

[70] Z. Yang, Y. Wang, C. Xu, P. Du, C. Xu, C. Xu, and Q. Tian, “Discernible Image Compression”, in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1561–1569.

[71] N. Le, H. Zhang, F. Cricri, R. Ghaznavi-Youvalari, and E. Rahtu, “Image Coding For Machines: an End-To-End Learned Approach”, in *ICASSP*, 2021, pp. 1590–1594.

[72] H. Choi and I. V. Bajić, “Scalable Image Coding for Humans and Machines”, *IEEE Transactions on Image Processing*, vol. 31, pp. 2739–2754, 2022.

[73] K. Liu, D. Liu, L. Li, N. Yan, and H. Li, “Semantics-to-Signal Scalable Image Compression with Learned Revertible Representations”, *IJCV*, vol. 129, no. 9, pp. 2605–2621, 2021.

[74] N. Yan, C. Gao, D. Liu, H. Li, L. Li, and F. Wu, “SSSIC: Semantics-to-Signal Scalable Image Coding With Learned Structural Representations”, *IEEE Transactions on Image Processing*, vol. 30, pp. 8939–8954, 2021.

[75] A. Harell, A. De Andrade, and I. V. Bajić, “Rate-Distortion in Image Coding for Machines”, in *Picture Coding Symposium*, 2022, pp. 199–203.

[76] Y. Bai, X. Yang, X. Liu, J. Jiang, Y. Wang, X. Ji, and W. Gao, “Towards End-to-End Image Compression and Analysis with Transformers”, in *AAAI*, vol. 36, 2022, pp. 104–112.

[77] M. I. Patel, S. Suthar, and J. Thakar, “Survey on Image Compression using Machine Learning and Deep Learning”, in *ICCS*, 2019, pp. 1103–1105.

[78] D. Mishra, S. K. Singh, and R. K. Singh, “Deep Architectures for Image Compression: A Critical Review”, *Signal Processing*, vol. 191, p. 108 346, 2022.

[79] M. Makar, H. Lakshman, V. Chandrasekhar, and B. Girod, “Gradient Preserving Quantization”, in *ICIP*, 2012, pp. 2505–2508.

[80] X. Xie and K.-H. Kim, “Source Compression with Bounded DNN Perception Loss for IoT Edge Computer Vision”, in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[81] K. Du, Q. Zhang, A. Arapin, H. Wang, Z. Xia, and J. Jiang, “AccMPEG: Optimizing Video Encoding for Video Analytics”, in *MLSys*, 2022.

[82] H. Itsumi, F. Beye, V. Charvi, and K. Nihei, “Learning Important Regions via Attention for Video Streaming on Cloud Robotics”, in *IROS*, 2022, pp. 6833–6839.

[83] E. Hamilton, “JPEG File Interchange Format”, 1992.

[84] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All you Need”, in *NeurIPS*, vol. 30, 2017.

[85] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going Deeper With Image Transformers", in *ICCV*, 2021, pp. 32–42.

[86] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, "Improving Deep Neural Networks Using Softplus Units", in *IJCNN*, 2015, pp. 1–4.

[87] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization", in *NIPS 2016 Deep Learning Symposium*, 2016.

[88] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)", *arXiv :1606.08415*, 2016.

[89] M. N. Rabe and C. Staats, "Self-attention Does Not Need $\mathcal{O}(n^2)$ Memory", *arXiv :2112.05682*, 2021.

[90] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation", in *ICLR*, 2018.

[91] Y. Tian, G. Lu, X. Min, Z. Che, G. Zhai, G. Guo, and Z. Gao, "Self-Conditioned Probabilistic Learning of Video Rescaling", in *ICCV*, 2021, pp. 4490–4499.

[92] A. Clark, *Pillow (PIL Fork) Documentation*, 2015. [Online]. Available: https://github.com/python-pillow/Pillow.

[93] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. M. Lee, "Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study", in *CVPRW*, 2019.

[94] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press.

[95] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection", in *ICCV*, 2017, pp. 2980–2988.

[96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", in *NeurIPS*, vol. 32, 2019.

[97] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey", *IJCV*, vol. 129, pp. 1789–1819, 2021.

[98] R. Hesse, S. Schaub-Meyer, and S. Roth, "FunnyBirds: A Synthetic Vision Dataset for a Part-Based Analysis of Explainable AI Methods", in *ICCV*, 2023, pp. 3981–3991.

[99] G. Erion, J. D. Janizek, P. Sturmfels, S. M. Lundberg, and S.-I. Lee, “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”, *Nature Machine Intelligence*, vol. 3, no. 7, pp. 620–631, 2021.

[100] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A ConvNet for the 2020s”, in *CVPR*, 2022, pp. 11 976–11 986.

[101] C. E. Shannon, “Communication in the Presence of Noise”, *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[102] R. Zhang, “Making Convolutional Networks Shift-Invariant Again”, in *ICML*, 2019, pp. 7324–7334.

[103] J. Grabinski, J. Keuper, and M. Keuper, “Aliasing and adversarial robust generalization of CNNs”, *Machine Learning*, vol. 111, no. 11, pp. 3925–3951, 2022.

[104] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu koray, R. Munos, and M. Valko, “Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning”, in *NeurIPS*, vol. 33, 2020, pp. 21 271–21 284.

[105] P. H. Richemond, J.-B. Grill, F. Altché, C. Tallec, F. Strub, A. Brock, S. Smith, S. De, R. Pascanu, B. Piot, and M. Valko, “BYOL works even without batch statistics”, *arXiv:2010.10241*, 2020.

[106] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, “SwinIR: Image Restoration Using Swin Transformer”, in *ICCVW*, 2021, pp. 1833–1844.

[107] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity”, *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[108] P. W. Glynn and R. Szechtman, “Some New Perspectives on the Method of Control Variates”, in *Monte Carlo and Quasi-Monte Carlo Methods 2000*, Springer, 2002, pp. 27–49.

[109] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, “Kornia: an Open Source Differentiable Computer Vision Library for PyTorch”, in *WACV*, 2020, pp. 3674–3683.

[110] TorchVision maintainers and contributors, *TorchVision: PyTorch’s Computer Vision library*, https://github.com/pytorch/vision, 2016.

[111] I. Loshchilov and F. Hutter, “SGDR: Stochastic Gradient Descent with Warm Restarts”, in *ICLR*, 2017.

[112] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, in *ICLR*, 2015.

[113] W. A. Falcon, “Pytorch Lightning”, *GitHub*, vol. 3, 2019.

[114] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, in *ICLR*, 2015.

[115] R. Jonschkowski, A. Stone, J. T. Barron, A. Gordon, K. Konolige, and A. Angelova, “What Matters in Unsupervised Optical Flow”, in *ECCV*, 2020, pp. 557–572.

[116] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum Learning”, in *ICML*, 2009, pp. 41–48.

[117] X. Wang, Y. Chen, and W. Zhu, “A Survey on Curriculum Learning”, *IEEE PAMI*, vol. 44, no. 9, pp. 4555–4576, 2021.

[118] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, “Curriculum Learning: A Survey”, *IJCV*, vol. 130, no. 6, pp. 1526–1565, 2022.

[119] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN”, in *CVPR*, 2020, pp. 8110–8119.

[120] N. Park and S. Kim, “How Do Vision Transformers Work?”, in *ICLR*, 2022.

List of Figures

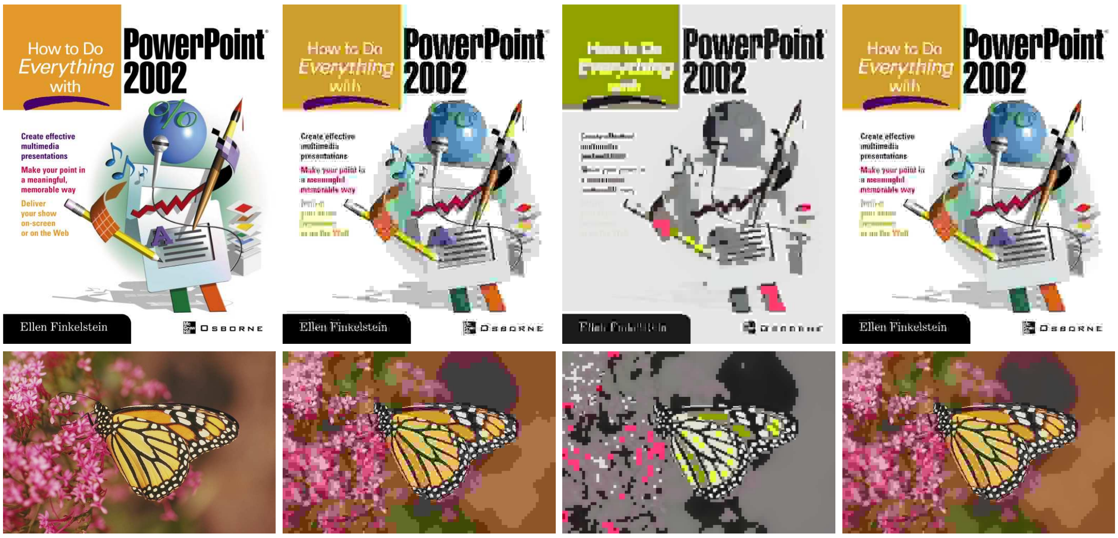
List of Tables

List of Algorithms

A. Appendix

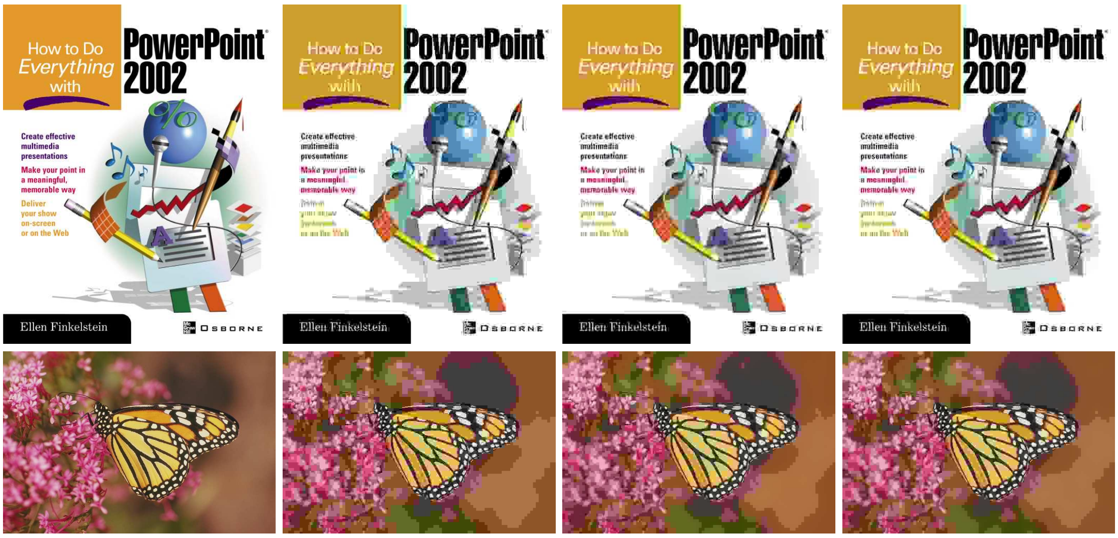
Here we report additional observations and results of the proposed methodology.

A.1. Additional Differentiable JPEG Coding Results


How to Do Everything with
PowerPoint 2002
Create effective multimedia presentations
Make your point in a meaningful, memorable way
Deliver your show on-screen or on the Web
Ellen Finkelstein
OSBORNE

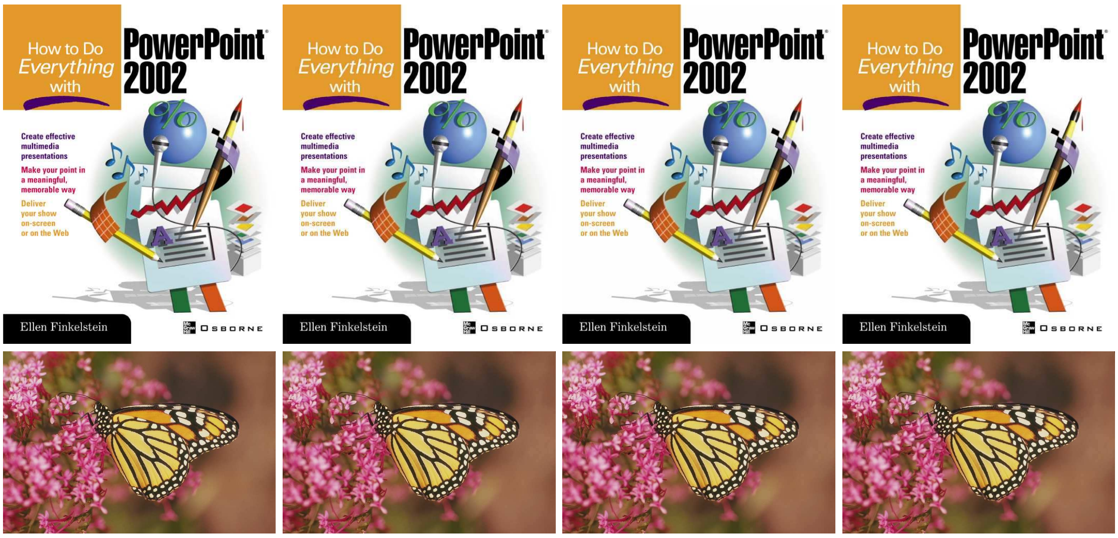
(a) Original image (b) JPEG (OpenCV) (c) Diff. JPEG (Shin *et al.*) (d) Diff. JPEG (ours)

Figure A.1. **Differentiable JPEG coding results (I) Utilized JPEG quality is 1.**


How to Do Everything with
PowerPoint 2002
Create effective multimedia presentations
Make your point in a meaningful, memorable way
Deliver your show on-screen or on the Web
Ellen Finkelstein
OSBORNE

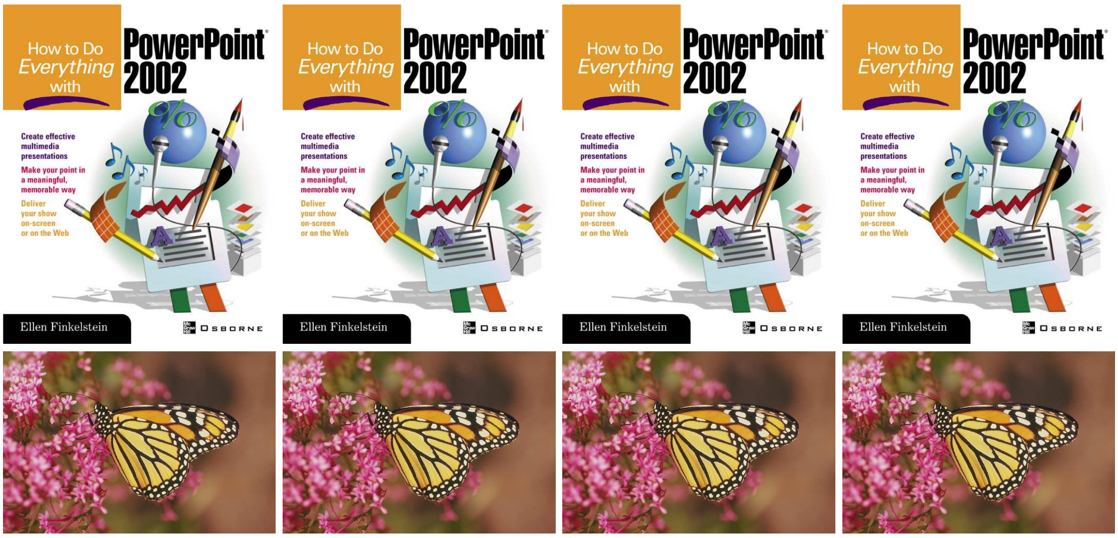
(a) Original image (b) JPEG (OpenCV) (c) JPEG (Shin *et al.*) (d) JPEG (ours)

Figure A.2. **Differentiable JPEG coding results (II) Utilized JPEG quality is 3.**


How to Do
Everything
with
PowerPoint
2002
Create effective multimedia presentations
Make your point in a meaningful, memorable way
Deliver your show on-screen or on the Web
Ellen Finkelstein
OSBORNE

(a) Original image (b) JPEG (OpenCV) (c) JPEG (Shin *et al.*) (d) JPEG (ours)

Figure A.3. **Differentiable JPEG coding results (III) Utilized JPEG quality is** 50**.**



(a) Original image (b) JPEG (OpenCV) (c) JPEG (Shin *et al.*) (d) JPEG (ours)

Figure A.4. **Differentiable JPEG coding results (IV) Utilized JPEG quality is 99.**

A.2. Differentiable JPEG File Size Preliminary Results

During preliminary we tested different approaches to incorporate positional encodings to the differentiable JPEG file size model. We experimented with both adding the sinusoidal positional encodings and concatenating the positional encodings to the quantized DCT features. Interestingly, we observed that concatenating the positional encodings leads to a way smoother and faster convergence. This was especially notable during the early phase of the training as showcased in Fig. A.5. Note we also experimented with no positional encodings at all. While this leads to a similarly fast initial convergence training as concatenating the positional encodings but leads to a notably more unstable training and weaker overall performance since the model lacks positional information.

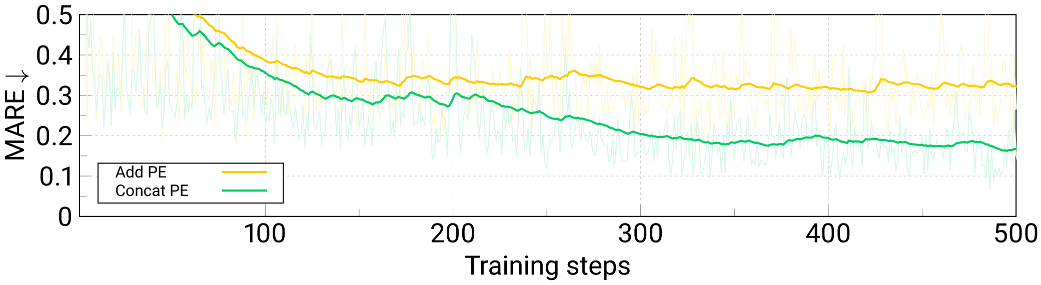


Figure A.5. **Add *vs.* concatenate positional encodings.** Training MARE at the beginning of the training. Concatenating the positional encodings (in green 🟩) leads to a better initial convergence than adding the positional encodings (in yellow 🟨) to the quantized DCT features. We showcase the exponential ruining average of the training MARE.