




Estimating the resize parameter in end-to-end learned image compression

Li-Heng Chen ^a ,* Christos G. Bampis ^b, Zhi Li ^b, Lukáš Krasula ^b, Alan C. Bovik ^a

^a Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA

^b Netflix, Inc, Los Gatos, CA, USA

ARTICLE INFO

Keywords:

Resizing
Convolutional neural networks
Learned image compression

ABSTRACT

We describe a search-free resizing framework that can further improve the rate–distortion tradeoff of recent learned image compression models. Our approach is simple: compose a pair of differentiable downsampling/upsampling layers that sandwich a neural compression model. To determine resize factors for different inputs, we utilize another neural network jointly trained with the compression model, with the end goal of minimizing the rate–distortion objective. Our results suggest that “compression friendly” downsampled representations can be quickly determined during encoding by using an auxiliary network and differentiable image warping. By conducting extensive experimental tests on existing deep image compression models, we show results that our new resizing parameter estimation framework can provide Bjøntegaard-Delta rate (BD-rate) improvement of about 10% against leading perceptual quality engines. We also carried out a subjective quality study, the results of which show that our new approach yields favorable compressed images. To facilitate reproducible research in this direction, the implementation used in this paper is being made freely available online at: <https://github.com/treammm/ResizeCompression>.

1. Introduction

When properly applied, spatial resolution reduction is a simple and natural way to compress visual information. It plays a significant role in many multimedia compression standards and applications. For example, it is common to decimate the chroma components of an image or video to reduce bitrate consumption before encoding a video, then upsize them before display. Many studies [1,2] have concluded that this fixed kind of sub-sampling operation results in little impact on perceived quality, because of the smaller bandwidth of chromatic information, and human perception of it [3]. In recent streaming video workflows, non-normative adaptive spatial resolution changes of both luma and chroma are now widely used by service providers like Netflix and Youtube to improve bandwidth consumption while maintaining the quality of experience of viewers [4–6]. Moreover, recent advanced video coding standards, such as AOMedia Video 1 (AV1) [7] and Versatile Video Coding (VVC) [8], have adopted the concept of spatial resizing as an in-loop coding tool [9,10]. Despite its usefulness, an important barrier to optimizing video resizing in compression workflows is the lack of an automatic protocol for finding the best reduced resolution in the perceptual rate–distortion sense. Current methods rely on exhaustive search over a set of resolutions.

In parallel with the recent successes of deep learning on many video processing problems [11–14], a number of promising learning-based

image and video compression models have been realized over the past few years. Unlike traditional hybrid codecs like H.264/AVC, which heavily rely on pipelines of hand-designed modules, most learned compression models deployed deep autoencoders that are optimized end-to-end on large datasets. A significant amount of research has been directed towards growing compression model capability and capacity, by modifying network architecture or by adopting detailed entropy models. The idea of incorporating directed spatial resolution changes into learned compression models has not received much attention, despite successes attained on conventional video coding.

Our aim here is to explore the potential of leveraging spatial resizing in learned lossy codecs. To overcome the aforementioned cost of computationally inefficient searches over the space of resize factors, we propose a framework that simultaneously learns the compression model and an estimate of the resize factor. As depicted in Fig. 1, the main idea is to regress on the resize factor M using an auxiliary neural network, which we will refer to as ResizeParamNet. Since the source image is the input to the auxiliary network, the resize factor M is estimated in a content-adaptive fashion, without the need for time-consuming search procedures. The input source image is downsampled before compression, then upsampled during reconstruction. Both resizing modules are controlled by the factor M . Our approach enables better rate–distortion performance without modifying the compression kernel.

* Corresponding author.

E-mail address: lhchen@utexas.edu (L.-H. Chen).

<https://doi.org/10.1016/j.image.2025.117277>

Received 22 August 2022; Received in revised form 26 April 2024; Accepted 9 January 2025

Available online 6 February 2025

0923-5965/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

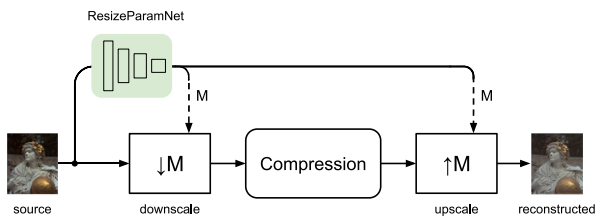


Fig. 1. Proposed framework for enhancing *learned* image compression using adaptive rescaling: additional downsampling ($\downarrow M$) and upsampling ($\uparrow M$) blocks are added before and after the compression network, respectively. The parameter $M \in (0, 1]$ is the scale factor, which is estimated by an auxiliary network trained in an end-to-end manner. Note that M is also signaled in the encoded bitstream.

We summarize the key characteristics and contribution of this work as follows:

- **Generalizability:** We show a way to inject optimized capability as a “booster” for learned image compression. This model, which we call *Resize-Compress*, can be used to jointly optimize differentiable resizing layers and a content-dependent resize factor estimator, and can be used with any compression models.
- **Efficiency:** Instead of using brute-force search, an auxiliary lightweight CNN (*ResizeParamNet*) is employed at the encoder side to quickly determine an optimal resize factor for a given input image. The estimated resize parameter is then signaled to the bitstream, with minimal bit overhead.
- **(Perceptual) Efficacy:** We comprehensively validate our proposed approach on a variety of representative deep image compression models, demonstrating that it leads to significant improvements in RD performance as measured by a variety of quality metrics. We also conducted a subjective picture quality study to further justify the perceptual relevance of our results.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 unfolds the details of our proposed resizing framework for learning-based compression models. Experiments and analysis are given in Section 4. Finally, Section 5 concludes the paper and draws future possible directions of this research.

2. Background

2.1. Learning-based lossy image and video compression

Recent years have witnessed a great surge of invention in the design of lossy image compression models realized by deep autoencoder architectures. They have been shown to achieve performances competitive with classical image coding standards, including JPEG2000 and HEVC Intra. Early efforts [15,16] focused on tackling fundamental challenges like non-differential quantization in order to construct end-to-end trainable infrastructures. Unlike other image-to-image transformation models, which mainly focus on reducing distortion, the bitrate is also approximated and taken into account during training. Later, more sophisticated designs were explored to extract increasingly compact features, or to reconstruct images with higher quality from compressed latent representations. For example, some recent approaches have adopted recurrent neural networks (RNNs) [17–19] to recursively compress residual information. Generative adversarial network (GANs) based compression models [20–22] produce reconstructed images whose statistics resemble the ground truth distributions. Although they often do not perform well with respect to pointwise quality measures, they can produce perceptually pleasing outcomes, especially at very low bitrates. Other methods like multi-scale networks [23] and invertible wavelet structures [24] have also been explored.

Another popular research direction has focussed on improving entropy estimation, which directly affects the bits that are required in

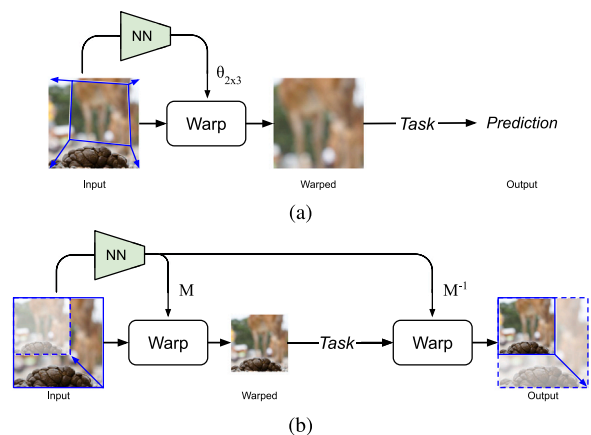


Fig. 2. Comparison of (a) the spatial transformer network (STN) [39] and (b) our framework of a pair of resizing layers for the compression task. The solid blue boxes indicate the warp of the input.

rate–distortion optimization. In [25], a scale hyperprior is introduced into the compression model. The authors use an additional network to estimate the standard deviation, to better model the conditional probability of the latent representation. Minnen et al. [26] and Lee et al. [27] further incorporate context-adaptive models to reduce local redundancies. These context models are often realized by PixelCNN-like modules. Other solutions have used 3D-CNNs to facilitate conditional probability modeling [28,29], or have adopted Gaussian mixture models [30] to improve likelihood estimation.

Beyond still pictures, considerable progress has also been made extending these ideas to video compression. Early attempts on end-to-end video compression, such as [31,32], have employed *frame interpolation* schemes which temporally interpolate frames in a video using neural networks. Residuals between source and interpolated frames are then encoded. Motivated by the motion estimation and motion compensation (MEMC) scheme in hybrid video codecs, a series of *optical flow*-based methods have also been proposed. For example, the DVC model [33] uses a pretrained optical flow model to provide temporal predictions. Rippel et al. [34] generalized the motion estimation process to achieve more efficient latent representations, and to compress motion and residuals simultaneously with a single autoencoder. In [35], a pretrained optical flow estimation model was combined with a learned encoder–decoder pair to perform bidirectional inter frame interpolation. Other variants of the optical flow setting, such as modeling motion using scale-space flow [36] or multi-scale flow [37], have been proposed to obtain promising improvements in coding efficiency. Interestingly, instead of estimating and signaling motion explicitly, the MOVI-Codec [38] captures motion regularities from displaced frame differences, without conducting any motion search.

Unlike the approaches mentioned above, our framework is a straightforward resizing-based compression framework that can be applied on any kind of neural compression kernels. In fact, the idea behind our resize parameter estimator is conceptually inspired by Spatial Transformer Networks (STN) [39], which is a type of neural network that is able to learn spatial affine transform parameters between images. Despite sharing some similarities, we point out important differences between our approach and STNs in Fig. 2. The STN predicts a 2×3 affine transform matrix $\theta_{2 \times 3}$ that is used in a bilinear warping layer. Since STNs are commonly used in high-level computer vision tasks like image recognition, it often results in cropping out a trapezoidal region of interest from the input image. However, when applying this concept to the compression task, a scalar M is instead learned: the resize parameter. To ensure that the reconstruction has the same resolution as the input, another warping layer is constructed, and constrained as an inverse transform back to the original geometry.

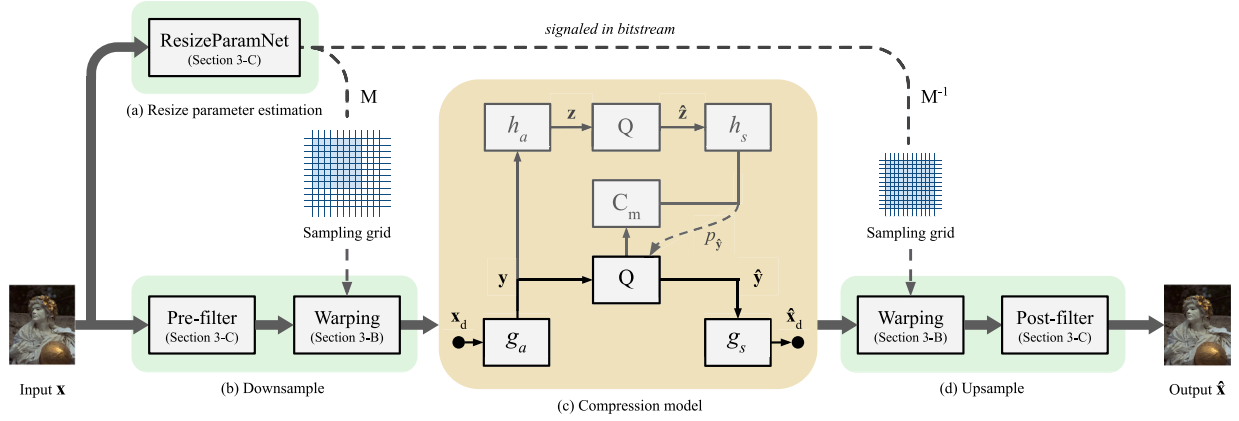


Fig. 3. Overview of our proposed Resize-Compress. First (a) estimate the resize factor M given an input source image. Using M , the image is downsampled and upsampled by module (b) and (d), respectively. Blocks shadowed in green are new components. Bold arrows indicate the flow of data in the framework, while thin dashed arrows represent the control signals being delivered to the resizing modules.

2.2. Resizing and its application in image and video coding

Unsurprisingly, the idea of applying spatial resizing in the context of conventional codecs has been deeply investigated and implemented in widespread practice. Typical use cases can be roughly categorized into the two classes. The first class includes codec-agnostic resizers operating in an *out-of-loop* manner, whereby a high-resolution source frame is spatially downsampled before encoding. An upscaling algorithm is implemented on the decoder side to scale the reconstructed frame back to its original resolution. Studies on image codecs like JPEG and JPEG2000 [40–42], and on video compression [43–45] have shown that encoding at lower resolutions generally results in better quality, when compressing to low bitrates. These schemes balance distortions produced by scaling against those by compression. Also in this class is a method often employed by streaming companies [4–6], where each source video is encoded at a finite set of combinations of resolution and compression levels, yielding multiple rate–distortion (RD) curves. Then, optimal encoding recipes are selected on the convex hull of the RD curves. This concept is simple, yet it operates at the extreme expense of exhaustive search, which is energy intensive and is also hard to apply in real-time applications. In efforts to avoid the need for search, more sophisticated resolution adaptation approaches have also been investigated [46–51].

The second class includes resizing incorporated as a normative *in-loop* coding tool. Shen et al. [52] proposed to encode inter frames at reduced resolutions. An example-based super resolution algorithm was designed to reconstruct high quality frames. The AV1 codec standard describes an option to horizontally scale a source frame to a lower resolution. Before updating reference frame buffers, linear upsampling is employed as part of an in-loop restoration filter [9,53]. In the standardization of VVC, an adaptive resolution change (ARC) scheme is defined [10] that enables inter prediction between frames having different resolutions. Improvements have been made on corresponding coding tools [54,55] to further improve coding efficiency.

3. Proposed method

Next we provide an overview of the Resize-Compress framework. We introduce the design methodology for each component, and analyze the learned resize factor both quantitatively and qualitatively. Finally, we present details on training and implementation.

3.1. Overview

Our proposed end-to-end trainable framework for resizing and learned compression is depicted in Fig. 3. Given an uncompressed

source image \mathbf{x} , the ResizeParamNet network first maps \mathbf{x} into a parameter M , the resize factor for the downsample module in Fig. 3(b). Accordingly, the estimated parameter M is used to generate a sampling grid \mathcal{T}_M , which is a set of fractional coordinates where the input should be sampled to produce the resampled output. To mitigate the information loss introduced by subsampling, a learnable convolutional layer (Pre-filter) is placed beforehand, allowing the retention of information. This is quite analogous to the precoding technique [56] used in communication systems, where channel information is coded on the input signal.

The image that has been downsampled by the warping layer, denoted by \mathbf{x}_d , is then encoded by a deep compression model. A general compression system (Fig. 3(c)) comprises an analysis transform g_a at the encoder side, and a synthesis transform g_s at the decoder side. The latent representation \mathbf{y} is first generated by applying the transformation g_a on the input

$$\mathbf{y} = g_a(\mathbf{x}_d). \quad (1)$$

Then, \mathbf{y} is quantized by the module Q and synthesized back by g_s , yielding the reconstructed image

$$\hat{\mathbf{x}}_d = g_s(\hat{\mathbf{y}}) = g_s(Q(\mathbf{y})). \quad (2)$$

Other modules such as a hyperprior (h_a, h_s) [25] or context model (C_m) [26,27] could also be utilized to obtain better estimates of the parameterized probability distribution. It is worth mentioning that the quantized representations (e.g., $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$) are encoded as discrete-valued data into the bitstream using an arithmetic coder. Finally, $\hat{\mathbf{x}}$ is output by upsampling $\hat{\mathbf{x}}_d$ using a similar warping layer (Fig. 3(d)), but with a reciprocal scaling parameter M^{-1} . Unlike the downsampling module in Fig. 3(b), the post-filtering network is placed *after* the warping layer to extract features that can repair the spatially degraded image. Similar methods have worked well in image super-resolution architectures [57].

3.2. Differentiable resize layer

It has been shown [39] that operations involving backward image warping with interpolation kernel $k(\cdot)$ are (sub-)differentiable with respect to all the arguments, allowing for the gradients to be back-propagated through the forward model. Inspired by this, we implement our resizing layers as a constrained form of image warping. We follow the convention in [39] to define a parameterized sampling grid \mathcal{T}_M for the i th pixel, which is a spatial affine coordinate transformation on the target coordinates of the warping output $(x'_i, y'_i)^T$

$$\mathcal{T}_M = \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{bmatrix} \frac{1}{M} & 0 \\ 0 & \frac{1}{M} \end{bmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \quad \forall i \in [1, \dots, HW], \quad (3)$$

where $(x_i^s, y_i^s)^T$ denote the corresponding input coordinates that define the sample points for interpolation. Under this scaling-constrained transformation, resolution is reduced when $0 \leq M < 1$, while $M > 1$ means an upscaling operation is performed. The output value of a particular pixel V_i located at $(x_i^t, y_i^t)^T$ can be written as

$$V_i = \sum_n \sum_m U_{mn} k(x_i^s - m)(y_i^s - n), \quad (4)$$

where U_{mn} is the input pixel value at location $(n, m)^T$. Therefore, warping an image $I = [U_i]$ can be expressed by

$$\text{Warp}_{\mathcal{T}_{M,k}}(I) = \text{Warp}_{\mathcal{T}_{M,k}}([U_i]) = [V_i]. \quad (5)$$

Note that an unwanted boundary could be created during downsampling, when the resampling grid $(x_i^s, y_i^s)^T$ exceeds the dimensions of the input source. In our implementation, we simply remove the unwanted boundary by cropping the warped image to $S \left\lceil \frac{\text{width}}{S} \right\rceil \times S \left\lceil \frac{\text{height}}{S} \right\rceil$, which is divisible by the equivalent stride of the compression model S .

In order to properly select the interpolation kernel k for our problem, we empirically conducted the following simplified tasks:

1. **Scale down:** An input $x = I$ is warped by a factor of M to fit the target image $y = \text{Warp}_{\mathcal{T}_{N,k}}(I)$, $N = 0.5$

$$f_1(x) = \text{Warp}_{\mathcal{T}_{M,k}}(x). \quad (6)$$

2. **Scale up:** A downsampled input $x = \text{Warp}_{\mathcal{T}_{N,k}}(I)$, $N = 0.5$ is warped by a factor of M^{-1} to fit the target image $y = I$

$$f_2(x) = \text{Warp}_{\mathcal{T}_{M^{-1},k}}(x). \quad (7)$$

3. **A resize pair:** An input x warped (downsampled) by a factor of M followed by an inverse warping ($x = y = I$)

$$f_3(x) = \text{Warp}_{\mathcal{T}_{M^{-1},k}}(\text{Warp}_{\mathcal{T}_{M,k}}(x)). \quad (8)$$

Note that N is a constant while M is a trainable variable, rather than the output of ResizeParamNet. For (6)–(8), we optimize M such that the MSE between $f_i(x)$ and y is minimized. Ideally, the value of M should converge to

$$\arg \min_M \|f(x) - y\|_2 = \begin{cases} 2^{-1}, & \text{if } f(x) = f_1(x) \\ 2^{-1}, & \text{if } f(x) = f_2(x) \\ 1, & \text{if } f(x) = f_3(x) \end{cases}. \quad (9)$$

These optimal parameters are intuitively determined for each task. For example, $M = 1$ minimizes the error for the third task defined in (8), since it is equivalent to an identity transformation, preserving the fidelity of the input. Fig. 4 compares the parameter M recorded during training with respect to the use of bilinear and bicubic interpolation. It may be observed that M quickly approached the optimal values in (9) when bicubic interpolation was used, whereas bilinear interpolation did not yield good convergence. This is likely due to the larger receptive field of the bicubic kernel, which results in stronger gradient signals for back propagation. Motivated by these observations, we implemented *bicubic* warping in our framework.

3.3. Network architecture

The details of the networks are outlined in Table 1, including the auxiliary module that is used to estimate resize factors, as well as the shallow filter networks placed at the two ends of the compression system.

The ResizeParamNet: The goal is to learn an $\mathbb{R}^{H \times W \times 3} \mapsto \mathbb{R}^1$ transformation that maps an input x to a resize factor M via a CNN. We constructed a network with similar design as the standard ResNet [58], consisting of three stages of residual blocks. The spatial size is reduced by a factor of 2 after each stage via 2×2 max pooling layers. Finally, 64 feature maps are fed to a global average pooling (GAP) layer, and the output is obtained by averaging the 64 values. The

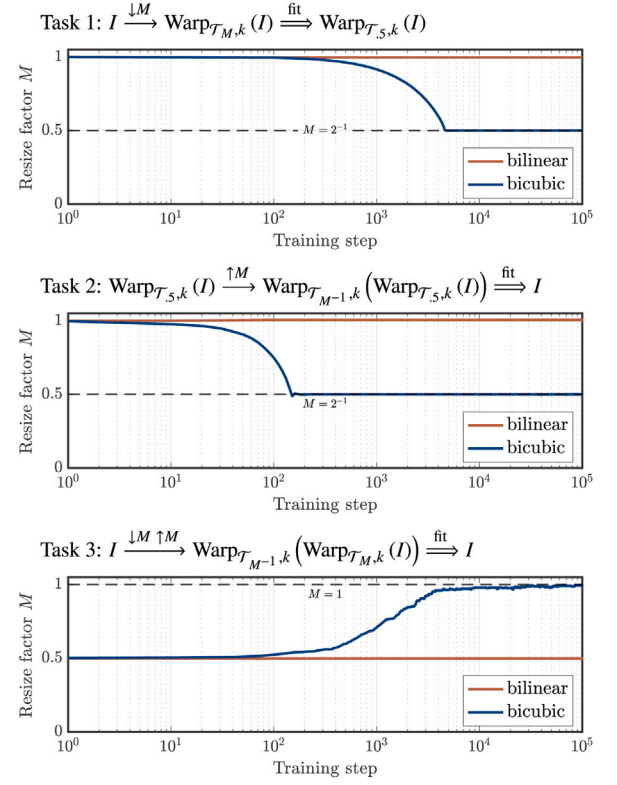


Fig. 4. Training plot of the predicted resize factor M using bilinear and bicubic interpolation kernels in different tasks. The black dashed lines denote the optimal values of M in each task as in (9).

Table 1

Architectural details of the additional networks in our framework.

	Name	Input	Operation	Activation	Output dim.
Pre/Post-filter	input	–	–	–	$W \times H \times 3$
	C1	input	Conv: $3 \times 3 c32 s1$	ReLU	$W \times H \times 32$
	C2	C1	Conv: $3 \times 3 c32 s1$	ReLU	$W \times H \times 32$
	C3	C2	Conv: $3 \times 3 c3 s1$	tanh	$W \times H \times 3$
	output	input, C3	Addition	–	$W \times H \times 3$
ResizeParamNet	input	–	–	–	$W \times H \times 3$
	R1	input	Res: $C = 16$	ReLU	$\frac{W}{2} \times \frac{H}{2} \times 16$
	R2	R1	Res: $C = 32$	ReLU	$\frac{W}{4} \times \frac{H}{4} \times 32$
	R3	R2	Res: $C = 64$	ReLU	$\frac{W}{8} \times \frac{H}{8} \times 64$
	P4	R3	Global Avg. Pooling	–	$1 \times 1 \times 64$
	output	P4	Mean	ReLU	1
Residual Block (Res)	input	–	parameters: C	–	$W \times H \times C_{in}$
	C1	input	Conv: $3 \times 3 cC s1$	–	$W \times H \times C$
	BN1	C1	Batch Norm.	ReLU	$W \times H \times C$
	C2	BN1	Conv: $3 \times 3 cC s1$	–	$W \times H \times C$
	BN2	C2	Batch Norm.	ReLU	$W \times H \times C$
	C3	input	Conv: $3 \times 3 cC s1$	–	$W \times H \times C$
	A4	BN2, C3	Addition	–	$W \times H \times C$
	output	A4	MaxPool: $2 \times 2 s2$	ReLU	$\frac{W}{2} \times \frac{H}{2} \times C$

Conv: convolutional layers denoted as kernel size|# of channels|stride.

MaxPool: max pooling layers denoted as pooling size| stride.

C_{in} : Number of input channels.

parameterization of each layer is detailed in the table. We experimented with different ways, such as using a fully connected (FC) layer, to aggregate the feature maps. However, we did not obtain improvements in performance.

The pre/post-filter networks: Both the *pre-filter* and the *post-filter* share the same network architecture of three stages of convolutional layers, accepting a 3-channel signal as input. The sizes of the convolutional layers are all fixed at 3×3 , while the number of filters is 32.

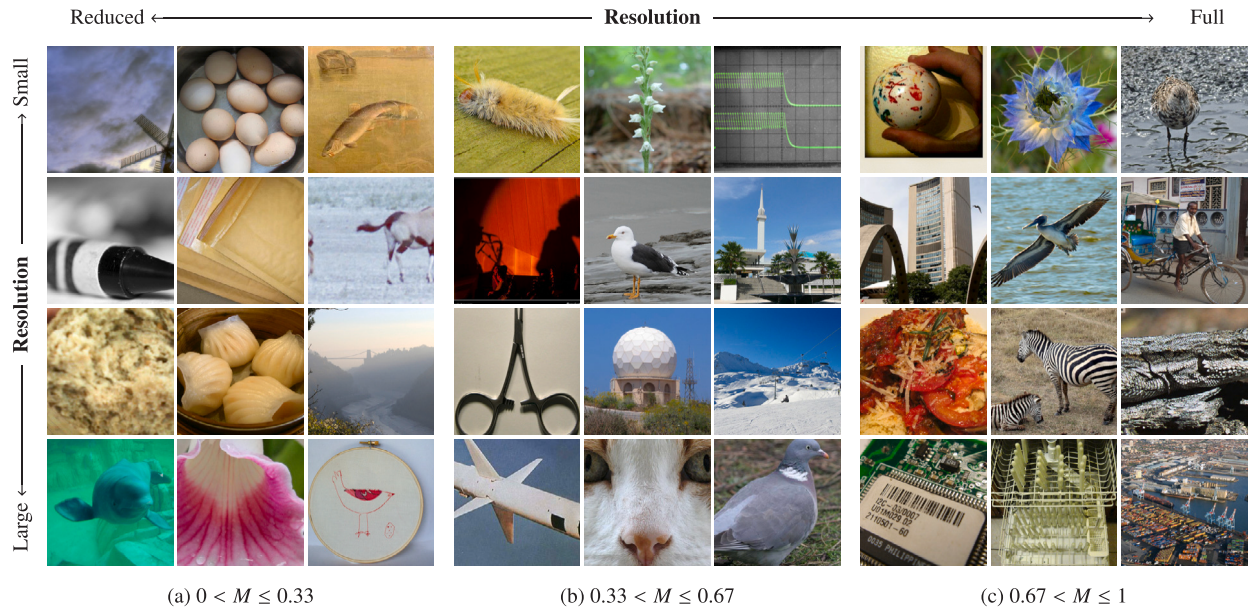


Fig. 5. Visualization of uncompressed *validation* patches associated with different resize factors M predicted by ResizeParamNet. ResizeParamNet was trained with the Ballé17 model [15] with $\lambda = 0.001$. These example patches were extracted from ImageNet.

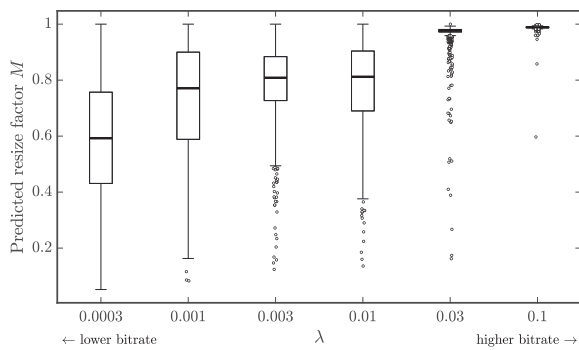


Fig. 6. Box plot of predicted resize factor M against the weighting parameter λ , evaluated on the *validation* patches extracted from ImageNet. The ‘o’ symbol denotes outliers. ResizeParamNet was trained with the Ballé17 model [15].

We zero pad the boundaries of the feature maps before applying each convolution, so that the output size is not reduced. Except for the last layer, all of the convolutional layers are activated by a ReLU nonlinearity. Finally, a 3-channel output is produced, yielding a residual that is added element-wise to the input image. It should be noted that the resolution changes produced by our model are handled by individual resize layers as described in Section 3.2, hence the stride parameter is always set to $s = 1$. We note that the two filters do not share parameters.

3.4. Loss function

Let θ_{g_a} , θ_{g_s} , ϕ_{pre} , ϕ_{post} , and ϕ_{resize} be the parameters of the analysis transform g_a , the synthesis transform g_s , the pre-filter, the post-filter, and ResizeParamNet, respectively. Our goal is to end-to-end optimize these parameters, such that the pipeline can generate a reconstructed batch \hat{x} that has high fidelity relative to the source batch x . Meanwhile, the cost of encoding quantized representations to bits should be as small as possible. Therefore, we train the model against the following losses. First, the distortion term measures the fidelity between the reconstructed image and its pristine source, which is defined as the residual between x and \hat{x} mapped by a distortion function d :

$$\mathcal{L}_{dist} = d(x - \hat{x}). \quad (10)$$

Here, the squared Euclidean distance $d(x) = \|x\|_2^2$ is adopted to minimize the mean squared error (MSE).

To representing the bit consumption of quantized latents, the rate loss is defined by

$$\mathcal{L}_{rate} = \sum_{\hat{q} \in Q} -\log_2 p_{\hat{q}}(\hat{q}), \quad (11)$$

where $p_{\hat{q}}$ refers to an entropy model estimated over the unknown distribution of input images, and Q denotes the set containing all the respective quantized latents. For example, in the work of Ballé et al. [25], the quantized latents \hat{y} and hyper-latents \hat{z} are encoded into bits, which is a case where $Q = \{\hat{y}, \hat{z}\}$. Coupling the losses from (10) and (11), and all the trainable parameters collectively denoted by $\theta = \{\theta_{g_a}, \theta_{g_s}, \phi_{pre}, \phi_{post}, \phi_{resize}, p_{\hat{q}} | \hat{q} \in Q\}$, the overall training objective that is used to optimize the system is defined as:

$$\mathcal{L}_{total}(\theta) = \lambda \mathcal{L}_{dist} + \mathcal{L}_{rate}, \quad (12)$$

where λ is a weight parameter that balances bitrate against distortion of the encoded bitstream. By increasing λ , better quality of reconstructed images can be achieved at the cost of compression ratio. The loss derivative was used to update the model parameters θ by back-propagating through the feed forward model. It is also interesting to note that (12) does not contain any loss terms specific to the resize factor M . Instead, the networks learn to directly estimate the resize factor as a byproduct of minimizing the rate-distortion loss.

3.5. Visualization of the learned resizing factors

To understand how a source image is resized in our framework, we used a subset of 1000 image patches from the ImageNet database, and fed them into ResizeParamNet trained with different weighting parameters λ . In Fig. 5, exemplar patches are arranged according to the estimated resize factor M to demonstrate the efficacy of ResizeParamNet. Generally, contents containing significant detail and texture are assigned values of M closer to 1, indicating less resolution reduction will be applied before compression. This is not unexpected, since high-frequency components are more susceptible to degradation from scaling. On the other hand, the patches of less complexity, or that are more blurry, are compressed at lower resolutions using smaller values of M . This is because in such instances, bitrate consumption can be

Table 2

Overall comparison of different codecs and results of optimized deep image compression on three datasets. Each cell shows the average change of BD-rate expressed as percent. The baseline comparison is against the Ballé17 model [15]. Smaller or negative values indicate better coding efficiency.

Image dataset	KODAK [59]				TECNICK [60]				JPEG AI [61]			
	PSNR	MS-SSIM	VIF	VMAF	PSNR	MS-SSIM	VIF	VMAF	PSNR	MS-SSIM	VIF	VMAF
JPEG	+181.69	+193.13	+207.60	+90.27	+173.98	+202.28	+179.30	+103.14	+167.29	+184.31	+192.59	+85.44
JPEG2000	-12.70	+0.07	+3.81	-33.54	-13.95	-9.92	-9.02	-34.20	-17.79	-11.69	-10.53	-35.80
WebP	-0.80	+2.72	+20.56	-18.90	+11.24	-2.67	+18.21	-15.78	+1.41	-0.94	+18.31	-16.40
HEVC Intra	-31.34	-7.44	-17.22	-30.39	-31.07	-13.79	-23.73	-29.43	-31.52	-12.01	-24.40	-25.62
Ballé17 [15] (Baseline)	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00	+0.00
Ballé17 +Resize (Ours)	-5.20	-6.60	-8.53	-10.25	-9.74	-14.40	-14.62	-12.91	-11.21	-11.44	-14.13	-10.60
Ballé18-Fact [25]	-11.52	-3.92	-2.29	-12.53	-8.52	-6.70	-4.34	-14.30	-9.94	-4.62	-4.31	-12.24
Ballé18-Fact +Resize (Ours)	-11.46	-7.56	-9.60	-17.84	-12.51	-17.67	-15.60	-20.09	-14.15	-15.12	-16.00	-19.03
Ballé18-Hyper [25]	-25.81	-13.07	-12.87	-31.08	-28.43	-21.45	-21.45	-35.89	-28.11	-19.51	-17.25	-33.52
Ballé18-Hyper +Resize (Ours)	-24.91	-19.69	-23.88	-42.93	-31.69	-32.44	-33.74	-45.81	-32.45	-29.76	-33.89	-44.21
Cheng20 [30]	-50.78	-42.95	-45.39	-56.20	-50.87	-48.75	-47.19	-58.28	-47.35	-43.57	-46.66	-54.58
Cheng20 +Resize (Ours)	-47.69	-44.66	-47.38	-58.00	-50.34	-49.45	-51.20	-59.30	-49.07	-44.69	-48.80	-55.65

significantly reduced by downsampling, while still maintaining similar quality level.

Fig. 6 plots the spreads of the predicted resize factor for models trained with different values of λ . It is evident that the estimated resize factor M increases with λ , as should be expected, since the loss function in (12) is dominated by the MSE term, making downsampling an undesirable operation. At the high-bitrate extreme of $\lambda = 0.1$, the distribution of M is centered around 1. As we will demonstrate in Section 4.2, this phenomenon is the underlying reason why the performance of our model reaches a saturation point in the high bitrate region. It is also worth noting that M spans a wide range of values when $\lambda \leq 0.001$. Similar to the observation we made in Fig. 5, this indicates the complex interplay between picture content, bitrate, and distortions: under constrained bitrate conditions, there is room to improve the (perceptual) rate–distortion tradeoff, by optimally applying different amounts of image resizing.

3.6. Implementation and training details

We developed and trained our models in Python using the TensorFlow framework (version 1.15). All of the models were trained using NVIDIA Tesla K80 GPU cards. We used the CLIC Professional Dataset [62], an image dataset consisting of high quality pictures, as training data. We did not use images smaller than 384 pixels along either the vertical or horizontal axis, resulting in a subset of 1600 images. No augmentation was applied on the training data.

During training, we used the Adam solver [63] to optimize the networks, with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a batch size of 8 with a crop size of 256 pixels. The networks were initially trained over 1M iterations, using a learning rate that was fixed at $1e^{-4}$. Then, the learning rates were decreased to $3e^{-5}$ for an additional 1M iterations. We found that using larger patches tended to facilitate the training stability of our framework, but with much slower training speed. Thus, we further refined the models using patches of size 384×384 for only 100K steps, resulting in a total of 2.1M iterations of backpropagation. For fair comparison, all of the models (including the original baselines) were trained from scratch under the same conditions.

4. Experiments and analysis

4.1. Evaluation experiments

Evaluation Dataset To compare our method and various image codecs, we utilized the well-known Kodak dataset [59] of 24 very high quality uncompressed 768×512 images. This publicly available image set is commonly used to evaluate image processing algorithms. We also used the Tecnick dataset [60] containing 100 images of 1200×1200 resolution, and 16 test images that are nearly ultra high definition from the JPEG-AI Call for Evidence [61], yielding images having more

diverse resolutions and contents. None of the test images were included in the training sets, to avoid bias and overfitting problems.

Evaluation Protocol The coding efficiency on the test set was measured by the Bjøntegaard-Delta bitrate (BD-rate) [64] of each image codec, which quantifies average differences in bitrate at a same distortion level relative to another reference encode. A negative BD-rate means that the bitrate was reduced as compared with the baseline. We encoded the images at 8 approximate different bitrates, ranging from 0.05 bpp (bits per pixel) to 1 bpp. Then, the BD-rates with respect to a variety of quality models were calculated. Aside from measuring the pixel-wise PSNR for completeness, we mainly relied on perception-based quality models, including MS-SSIM [65], VIF [66] and VMAF [67], to quantify the distortion levels that were used for BD-rate calculation. We are aware of recent studies [68,69] of quality evaluation on deep learning based image compression. It has been shown that in this context, these three perceptual quality models have the highest correlation against subjective scores, whereas absolute fidelity models like the PSNR correlate poorly with visual perception, producing significantly inferior quality predictions than perception-based quality predictors. It is also worth noting that PSNR was not even considered as an evaluation criteria in the JPEG-AI Call for Evidence [61].

4.2. Overall comparison

We comprehensively evaluated neural compression models against four conventional image codecs: JPEG, JPEG2000, WebP, and the intra-coded HEVC main-RExt (Format Range Extension) profile. For all conventional codecs, we choose maximum gain over encoding speed, and conducted encoding in the codec's native YCbCr color space without chroma subsampling. Extensive experiments were carried out on the three aforementioned datasets, using four representative deep compression models as the backbone to test the generality of our framework: Ballé17¹ [15], Ballé18-Fact¹ (the model using a factorized prior in [25]), Ballé18-Hyper¹ (the hyperprior model proposed in [25]), and Cheng20² [30]. We tabulated the performances of all of the compared models in Table 2, with respect to different objective image quality assessment models. We report the BD-rate changes obtained relative to the baseline (Ballé17 [15]), averaged over all the images in the test set.

As may be observed from the results in the table, our approach (highlighted in **boldface**) was able to deliver significant BD-rate gains against the original compression model. Interestingly, greater RD performance gains achieved against MS-SSIM, VIF, and VMAF show the perceptual benefits of our proposed Resize-Compress framework to

¹ <https://github.com/tensorflow/compression>

² We implemented the training code based on the network architecture provided by the authors, which has been made available online at <https://github.com/treamm/TTrainCompression-ChengCVPR2020>.

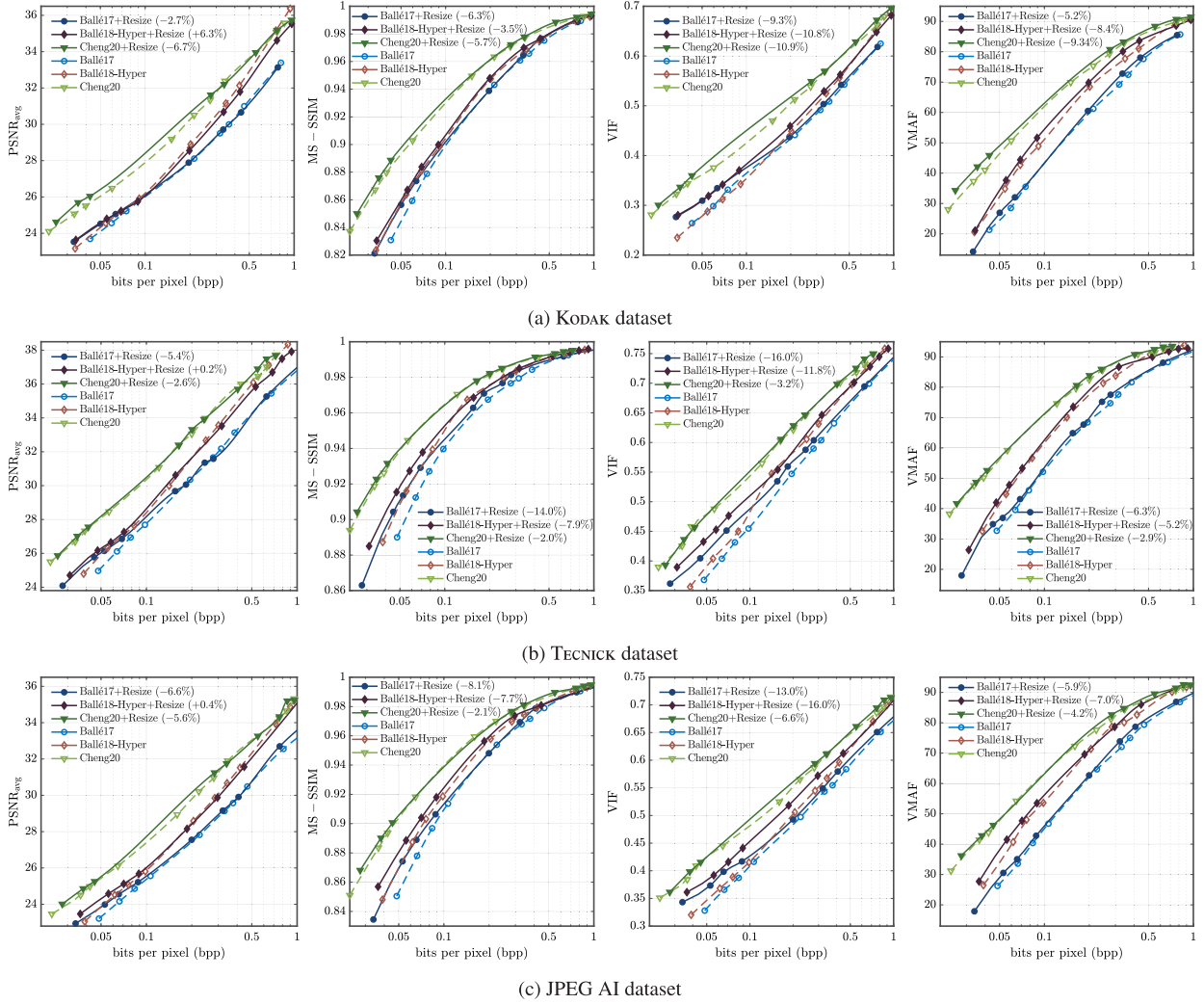


Fig. 7. Rate–distortion plots aggregated over of the (a) Kodak, (b) Tecnick, and (c) JPEG AI datasets. The numbers in parentheses are the BD-rates of our Resize-Compress framework against the original compression models. We display the x -axis on a log scale and excluded Ballé18-Fact for better visualization. The BD-rates shown in the figures were calculated from C +Resize against C , where C represents a compression model.

more general perceptual metrics, even though it was trained using MSE loss. For example, despite having similar or marginally worse PSNR BD-rates, we were able to demonstrate that building the early Ballé17 model on top of our framework (**Ballé17+Resize**) achieved similar or better results than the more complex Ballé18-Fact model, when the results were measured by the perceptual quality models. This suggests that the improvements made were not only in a pixel-wise (PSNR) sense, but that the ResizeParamNet model also learned to resize images that contributed favorably towards optimizing the visual quality of neural image compression models. It is worth noting that the PSNR BD-rates obtained using our framework performed better particularly on higher resolution test sets (Tecnick and JPEG-AI), possibly because large resolution images are less affected by the boundary cropping described in Section 3.2. In some cases, in particular the Kodak dataset, the BD-rate with respect to the PSNR metric did not always perform as well as the original compression model. This may have been a problem with PSNR, as we will analyze next.

In addition to the overall BD-rate results, we present the aggregated R–D curves on the three test datasets in Fig. 7 to study coding performance at different bitrates. In this scenario, each R–D point presents the aggregated value calculated by averaging across all test images compressed by the same model with a specific value of λ . To be clear, the calculation of R–D performance here is different from that of

Table 2, which followed common practice in the MPEG community of averaging over per-image BD-rates. Similar to the trends observed from Table 2, significant coding gains were obtained by applying our Resize-Compress framework, especially as measured by perceptual quality models. As is evident, the benefits obtained by resolution reduction are most significant in low bitrate compression scenarios (e.g., below 0.5 bpp), while the models with Resize-Compress delivered similar levels of performance at high bitrates. This is not surprising, and has been well demonstrated in early literature [40–42], since the reduction in quantization error is not large enough to offset the increase of distortion artifacts in high-quality compression. The dominance of the distortion in the loss function for the high bitrate encoding models also suggests a lower likelihood of bitrate reduction through downscaling. It may also be inferred from the figures that the worse PSNR BD-rate case in the low-resolution Kodak set was caused at high bitrates, likely due to the property of PSNR: when PSNR is large, it is more sensitive to small changes in the MSE. Since our models were trained on high resolution datasets, slight estimation errors of the resize factor may be introduced when testing on low resolution images. Consequently, a small increment in MSE may result in a noticeable decrement in PSNR. Yet, this did not affect perceived quality, as attested by the perceptual quality measurements and by the subjective study we will describe later.

Table 3

Ablation experiments on the design of resize layers and resize factor estimation. The baseline comparison is against the Ballé17+Resize model.

Ablation	PSNR	MS-SSIM	VIF	VMAF
Ballé17 +Resize (full model)	+0.00	+0.00	+0.00	+0.00
Ballé17 [15] (original model)	+1.48	+11.29	+13.93	+14.00
(a) w/o bicubic (w/ bilinear)	+2.29	+2.99	+1.21	+2.38
(b) w/o pre-filter	+0.02	+4.21	+3.90	+6.40
(c) w/o post-filter	+3.87	+8.31	+8.14	+16.70
(d) = (a) + (b) + (c)	+10.32	+13.87	+13.55	+22.98
(e) w/o ResizeParamNet	+0.87	+3.14	+8.23	+2.96
(f) w/o ResizeParamNet ($M = 1$)	+1.14	+8.64	+10.46	+11.75
(g) est. $\mathbf{M} = (M_x, M_y)$	+12.86	+2.62	+2.94	+5.48
(h) Lanczos interpolation	-0.38	+1.05	-1.18	-1.69

4.3. Ablation study

In order to study the significance of each module in our framework, we conducted an ablation study by training a series of intermediate models between the original Ballé17 model [15] and the corresponding using Resize-Compress. In this study, all the models were re-trained with 256×256 patch sizes and with the learning rate fixed at $1e^{-4}$ for 1M steps. For simplicity, we only tested the performance at four different bitrates, ranging from 0.025 bpp to 0.75 bpp. We measured changes the R-D performance introduced by removing components from the full model.

Study of resize layers We studied the design choice of resize layers by comparing the four intermediate models below against the full model:

- (a) Replacing bicubic interpolation in both down-sampling and up-sampling layer with bilinear interpolation;
- (b) Removing the pre-filter in the down-sampling layer;
- (c) Removing the post-filter in the up-sampling layer;
- (d) Removing the pre-filter and the post-filter on top of (a).

From the results in Table 3, we can draw a number of conclusions. The first thing to notice is the drop in performance attained by removing each component. As expected, the best performances were achieved using the full model, indicating that all the modules were important. It may also be observed that adding a post-filter in the up-sampling layer was more effective than using the pre-filter. Among the four cases, the poorest performance in (d) shows that, simply using bilinear interpolation to resize images is insufficient in this context.

Importance of resize parameter estimator Estimating parameters for image resizing is another essential component of Resize-Compress. In this experiment we investigated the significance of the estimation methodology against complexity:

- (e) Learning a content-agnostic resize factor using a trainable variable M (without ResizeParamNet);
- (f) Removing the ResizeParamNet and the trainable variable M (fixed $M = 1$);

In (e), we found that the performance of the content-agnostic model underperformed the full model, since it lacked the flexibility to adjust resize parameter according to the input content. The result given by the experiment (f) demonstrated the importance of having resizing, since the perceptual rate-distortion performance dropped significantly. In other words, merely deepening the compression network with the pre-filter and post-filter is insufficient to improve the performance.

We also tried to improve the performance by adopting more complex models. Specifically,

- (g) Estimating separate resize parameters $\mathbf{M} = (M_x, M_y)$ for the x , y dimensions, respectively. Here M_x and M_y were predicted by two separate CNNs.

Table 4

Runtime comparison of deep compression models. All computational speeds are given in milliseconds.

Compression model	Enc. (CPU)	Dec. (CPU)	Enc. (GPU)	Dec. (GPU)
Ballé17 [15]	320.62	369.65	210.16	222.73
Ballé17 +Resize	1136.69	702.22	393.69	307.76
Ballé17 +Resize [†]	749.02	513.62	318.56	267.64
Ballé18-Fact [25]	824.52	1086.26	199.46	263.77
Ballé18-Fact +Resize	1686.69	1336.37	431.51	343.75
Ballé18-Fact +Resize [†]	1332.70	1147.26	377.16	310.15
Ballé18-Hyper [25]	1006.54	1096.46	308.52	301.09
Ballé18-Hyper +Resize	1781.32	1463.94	595.75	419.84
Ballé18-Hyper +Resize [†]	1416.68	1204.02	488.91	386.11
Cheng20 [30]	46 136.25	47 263.21	42 506.73	44 553.21
Cheng20+Resize	46 526.51	47 392.57	42 736.56	45 012.58
Cheng20+Resize [†]	46411.31	47305.02	42611.29	44892.03

Resize[†]: replace the customized differentiable bicubic interpolation layer by the built in Tensorflow bicubic resizing block (`tf.image.resize`), which is not differentiable to the resize factor M . Note that the output result may have small numerical difference due to the implementation.

- (h) Replacing bicubic interpolation in both down-sampling and up-sampling layer with Lanczos interpolation;

Surprisingly, the result in (g) shows that doubling the number of parameters to conduct independent bidirectional scaling significantly lowered the performance. This may have been due to training instability introduced by an overparameterized model. Thus, we recommend sharing the same resize parameter for both the x and y dimensions. In (h), the use of a more sophisticated Lanczos interpolation marginally improved the performance. This suggests that the pre-filter and post-filter were able to be trained to offset the gap between bicubic and Lanczos interpolations.

4.4. Execution time

We evaluated the processing times of the various compared deep image compression models in Table 4. The results were calculated by averaging the runtime (in terms of milliseconds) over all Kodak images on the same machine equipped with an Intel Xeon Platinum 8259CL CPUs@2.50 GHz, 128G RAM, and an NVIDIA Tesla K80 GPU. The model loading time was not included. From Table 4, it may be observed that, due to the integration of Resize-Compress, the encoding and decoding complexity was higher as compared to the original models, especially when measured on CPU. However, as seen in the Table, the complexity overhead of our framework was negligible on top of more sophisticated models like Cheng20. Since ResizeParamNet is not present during decoding, the decoding speed is slightly faster, and we may also infer that the customized differentiable bicubic interpolation is the component that slows the execution speed. This can be further optimized by using native languages such as C++ or by re-compiling with a low-level instruction set. In investigating this aspect, we experimented with substituting the custom bicubic layer with an optimized built-in resizing block (denoted by Resize[†]) during inferencing, which resulted in substantial improvements in runtime.

4.5. Subjective study and analysis

We conducted an online human subject study to better understand the perceptual preferences of human viewers against different compression models. We selected the Ballé17 and Ballé18-Hyper models for the study, and compared them against the corresponding versions equipped with our Resize-Compress framework. We adopted the *two-alternative forced choice (2AFC)* method, since it can be easier for humans to make comparisons between simultaneously displayed pictures having subtle perceptual differences at similar bitrates. We selected all 18 landscape image contents from the Kodak dataset, where each content was encoded at four different bitrates ranging from 0.01 bpp to 1 bpp. To

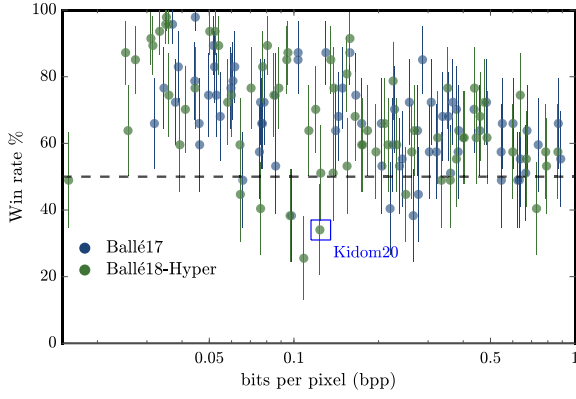


Fig. 8. Subjective performance of perceptual win rate of our Resize-Compress framework against the original compression model for each comparison (denoted by β). The error bars indicate the 95% binomial proportion confidence intervals obtained by $\hat{\beta} \pm 1.96\sqrt{\hat{\beta}(1-\hat{\beta})/n}$, $n = 47$.

equating bitrates in each comparison, we encoded the images using a 2D 13×13 grid of models at different λ values ($\lambda_{\text{original}}, \lambda_{\text{Resize-Compress}}$) $\in \{\lambda_i\} \times \{\lambda_i\}_{i=1,2,\dots,13}$, where $\lambda_i \in [0.0001, 0.1]$. We selected four λ pairs that minimized percent bitrate differences. We customized a user interface to carry out the human study, whereby participants could easily compare pairs of images in their natural environment. On each trial, subjects were shown two images encoded by the original compression model and our Resize-Compress framework, both presented next to the corresponding reference image. After viewing each image triplet, a subject was asked to choose which of the two images had better fidelity compared to the reference. The study included 47 volunteer participants and about 46% were somewhat knowledgeable about image/video processing, while the others were naive. In sum, we asked each participant to compare $18 \times 4 \times 2 = 144$ pairs of images, which lasted about 30-40 minutes.

Given the collected subjective comparisons, we analyzed the results by computing the percentage of subjects that preferred Resize-Compress over the original model, and plotted the results with respect to the bitrate of each comparison in Fig. 8. On average, the images encoded with Resize-Compress framework were preferred by 67.24% of the subjects. Among all the comparisons, 88 of 144 cases had win rates significantly larger than 50% while only 2 cases were worse. It may also be observed that, as bitrate was increased, our approach began to obtain win rates around 50% of the votes, since the distinctions between the codecs then becomes quite subtle.

Failure case analysis While Resize-Compress attained very good performance at low bitrate compression, it did not perform well in a few particular cases. Fig. 9 shows a failure case where more subjects rated the result without using our model as better (corresponding to the blue-boxed datapoint in Fig. 8). This may have been because the subjects were less forgiving of the blurred logo (highlighted by the red box) which may have worsened using our approach, and which may have drawn their attention. Conversely, the checkerboard artifacts (gray box) or blurry (yellow box) created by the original Ballé18-Hyper model may have been neglected due to their location and relative faintness. These cases further illustrate the challenges of predicting optimal resize factors for compression, given the content diversity of natural images.

2AFC similarity scores We are also interested in how different image quality assessment (IQA) models correlate with human perception in our study. A standard approach is to compare the correlations between human opinion scores and the predictions made by picture quality models. However, the number of comparisons we obtained in the study is insufficient to estimate the Mean Opinion Score (MOS) using the Bradley-Terry model [70]. To understand the performance of IQA models under this limitation, we instead adopted the 2AFC scoring

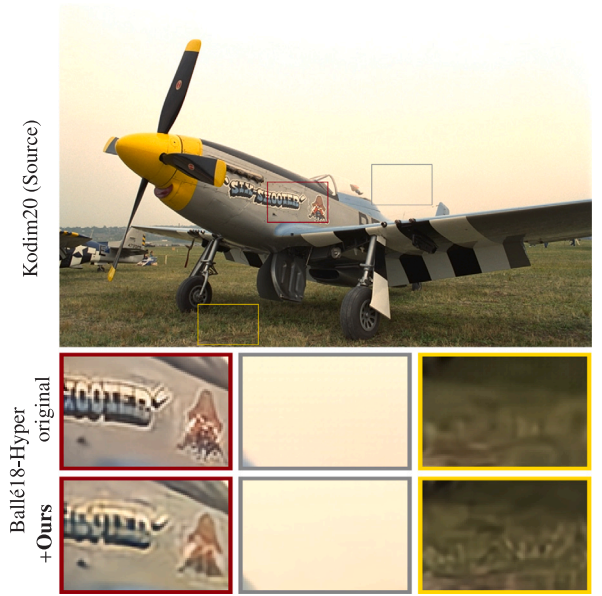


Fig. 9. A failure case on image Kodim20: the original Ballé18-Hyper at 0.135 bpp was preferred by 67% of the subjects over our framework (33% win rate) at 0.124 bpp.

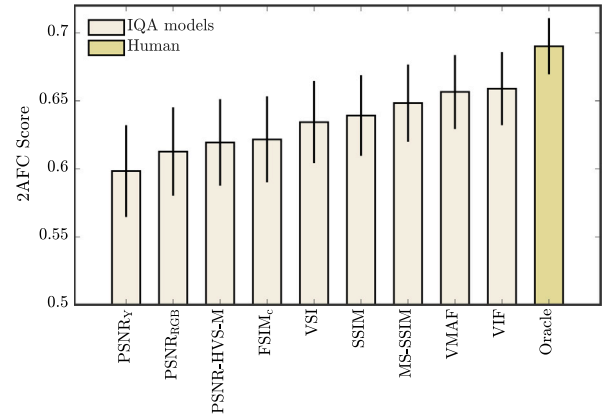


Fig. 10. Performance comparison of various IQA methods using the 2AFC score. Larger values indicate better agreement with human judgments.

method [71] given by $\hat{\beta}\hat{q} + (1-\hat{\beta})(1-\hat{q})$, where $\hat{\beta}$ is the percentage of human votes and $\hat{q} \in \{0, 1\}$ is the decision made by an IQA model. The idea behind this index is simple: when $\hat{\beta}$ agrees with \hat{q} , the 2AFC score is larger, indicating better performance of a quality model. The theoretical upper bound of the 2AFC score is given by $\max\{\hat{\beta}, 1-\hat{\beta}\}$, which can be achieved by an oracle agent $\hat{q} = \mathbb{1}_{\hat{\beta} > 0.5}$. We evaluated several popular image quality models, including SSIM [72], MS-SSIM [65], PSNR-HVS-M [73], VIF [66], VMAF [67], FSIM [74], and VSI [75], and plotted the 2AFC scores in Fig. 10. Overall, VIF and VMAF achieved the closest performance to the human oracle of 0.69, whereas MS-SSIM ranked a close third. On the other hand, the levels of performance attained by the PSNR family, which are commonly used in compression, were poor. Like other studies in the literature that analyzed the perceptual quality of learning based image compression models [68,69], our experiments also confirmed the value and perceptual relevance using models like MS-SSIM, VIF, and VMAF.

5. Conclusion and future work

We have introduced a framework for collectively optimizing resize parameter prediction and deep image compression. A distinguishing

characteristic of our design is that the optimal resize factor for different contents is estimated by an auxiliary network, without the need for pre-assigned labels. We experimentally demonstrated that, at low bitrate compression, our approach achieved significant improvements in terms of rate–distortion performance, relative to the original compression models. The RD performance gain measured with perceptual quality models and a subjective study further establish the efficacy of optimal resizing as a way of improving perceptual RD tradeoffs and subjective quality. The proposed learned resizing framework is simple and generalizes well across compression models while delivering significant performance improvements.

The concept of content-adaptive resizing may serve as a tractable and useful tool for perceptually improving the network architectures of other image restoration problems. Our results have also shown that the ResizeParamNet network produces powerful representations for resizing, with perceptual relevance. Looking ahead, it is possible that the deep features extracted from the learned ResizeParamNet network could be transferred as a “fast estimator” of optimal resize factors in the context of non-differentiable hybrid video codecs.

CRedit authorship contribution statement

Li-Heng Chen: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Validation, Visualization, Writing – original draft. **Christos G. Bampis:** Resources, Writing – review & editing. **Zhi Li:** Resources, Writing – review & editing. **Lukáš Krasula:** Resources, Writing – review & editing. **Alan C. Bovik:** Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank all the volunteers who took part in the subjective study. The human study was conducted under the approval of the Institutional Review Board (IRB) under protocol 2007-11-0066.

Data availability

We have shared github repository for this research, which includes the code and subjective study data.

References

- [1] S. Winkler, M. Kunt, C.J. van den Branden Lambrecht, Vision and video: Models and applications, *Vis. Model. Appl. Image Video Process.* (2001) 201–229.
- [2] M. Uhrina, J. Bienik, T. Mizdos, Chroma subsampling influence on the perceived video quality for compressed sequences in high resolutions, *Adv. Electr. Electron. Eng.* 15 (4) (2017).
- [3] K.T. Mullen, The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings, *J. Physiol.* 359 (1985) 381–400, <http://dx.doi.org/10.1113/jphysiol.1985.sp015591>.
- [4] A. Aaron, Z. Li, M. Manohara, J.D. Cock, D. Ronca, Per-title encode optimization, 2015, NETFLIX Tech Blog, URL <https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2>.
- [5] C. Chen, Y.-C. Lin, S. Bentsing, A. Kokaram, Optimized transcoding for large scale adaptive streaming using playback statistics, in: *Proc. IEEE Int. Conf. Image Process.*, 2018, pp. 3269–3273.
- [6] P.-H. Wu, V. Kondratenko, I. Katsavounidis, Fast encoding parameter selection for convex hull video encoding, in: *Proc. SPIE Applications Digital Image Process. XLIII*, 2020.
- [7] Y. Chen, D. Mukherjee, J. Han, A. Grange, Y. Xu, S. Parker, C. Chen, H. Su, U. Joshi, C.-H. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J.-M. Valin, T. Davies, S. Midtskogen, A. Norkin, P. de Rivaz, Z. Liu, An overview of coding tools in AV1: the first video codec from the alliance for open media, *APSIPA Trans. Signal Inf. Process.* 9 (2020).
- [8] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, J.-R. Ohm, Overview of the versatile video coding (VVC) standard and its applications, *IEEE Trans. Circuits Syst. Video Technol.* 31 (10) (2021) 3736–3764.
- [9] U. Joshi, D. Mukherjee, Y. Chen, S. Parker, A. Grange, In-loop frame super-resolution in AV1, in: *Proc. IEEE Picture Coding Symp.*, 2019.
- [10] Hendry, Y.-K. Wang, J. Chen, T. Davies, A. Fuldseth, Y.-C. Sun, T.-S. Chang, J. Lou, On adaptive resolution change (ARC) for VVC, 2019, Document JVET-M0135, ITU-T/ISO/IEC Joint Video Experts Team (JVET).
- [11] H.C. Burger, C.J. Schuler, S. Harmeling, Image denoising: Can plain neural networks compete with BM3D? in: *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2012, pp. 2392–2399, <http://dx.doi.org/10.1109/cvpr.2012.6247952>.
- [12] Y.-L. Liu, Y.-T. Liao, Y.-Y. Lin, Y.-Y. Chuang, Deep video frame interpolation using cyclic frame generation, in: *Proc. AAAI*, 2019, pp. 8794–8802.
- [13] S. Paul, A. Norkin, A.C. Bovik, Speeding up VP9 intra encoder with hierarchical deep learning-based partition prediction, *IEEE Trans. Image Process.* 29 (2020) 8134–8148, <http://dx.doi.org/10.1109/tip.2020.3011270>.
- [14] S. Paul, A. Norkin, A.C. Bovik, Self-supervised learning of perceptually optimized block motion estimates for video compression, 2021, ArXiv, abs/2110.01805.
- [15] J. Ballé, V. Laparra, E.P. Simoncelli, End-to-end optimized image compression, in: *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–27.
- [16] L. Theis, W. Shi, A. Cunningham, F. Huszár, Lossy image compression with compressive autoencoders, in: *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–19.
- [17] G. Toderici, S.M. O'Malley, S.J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, R. Sukthankar, Variable rate image compression with recurrent neural networks, 2015, CoRR abs/1511.06085.
- [18] G. Toderici, D. Vincent, N. Johnston, S.J. Hwang, D. Minnen, J. Shor, M. Covell, Full resolution image compression with recurrent neural networks, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 5306–5314.
- [19] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, G. Toderici, Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4385–4393.
- [20] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, L.V. Gool, Generative adversarial networks for extreme learned image compression, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 221–231.
- [21] J. Löhdefink, A. Bär, N.M. Schmidt, F. Hüger, P. Schlicht, T. Fingscheidt, GAN- vs. JPEG2000 image compression for distributed automotive perception: Higher peak SNR does not mean better semantic segmentation, 2019, ArXiv abs/1902.04311.
- [22] F. Mentzer, G.D. Toderici, M. Tschannen, E. Agustsson, High-fidelity generative image compression, in: *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 11913–11924.
- [23] K.M. Nakanishi, S. ichi Maeda, T. Miyato, D. Okanohara, Neural multi-scale image compression, in: *Proc. Asia. Conf. Comput. Vis.*, 2019, pp. 718–732.
- [24] H. Ma, D. Liu, N. Yan, H. Li, F. Wu, End-to-end optimized versatile image compression with wavelet-like transform, *IEEE Trans. Pattern Anal. Mach. Intell.* (2020) 1.
- [25] J. Ballé, D. Minnen, S. Singh, S.J. Hwang, N. Johnston, Variational image compression with a scale hyperprior, in: *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–23.
- [26] D. Minnen, J. Ballé, G.D. Toderici, Joint autoregressive and hierarchical priors for learned image compression, in: *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10771–10780.
- [27] J. Lee, S. Cho, S.-K. Beack, Context-adaptive entropy model for end-to-end optimized image compression, in: *Proc. Int. Conf. Learn. Represent.*, 2019.
- [28] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, L.V. Gool, Conditional probability models for deep image compression, in: *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2018, pp. 4394–4402.
- [29] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, Y. Wang, End-to-end learnt image compression via non-local attention optimization and improved context modeling, *IEEE Trans. Image Process.* 30 (2021) 3179–3191.
- [30] Z. Cheng, H. Sun, M. Takeuchi, J. Katto, Learned image compression with discretized Gaussian mixture likelihoods and attention modules, in: *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2020, pp. 7939–7948.
- [31] C.-Y. Wu, N. Singhal, P. Krähenbühl, Video compression through image interpolation, in: *Proc. Eur. Conf. Comput. Vis.*, 2018.
- [32] Z. Cheng, H. Sun, M. Takeuchi, J. Katto, Learning image and video compression through spatial-temporal energy compaction, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.
- [33] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, Z. Gao, DVC: An end-to-end deep video compression framework, in: *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 2019, pp. 11006–11015.
- [34] O. Rippel, S. Nair, C. Lew, S. Branson, A.G. Anderson, L. Bourdev, Learned video compression, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3454–3463.

- [35] A. Djelouah, J. Campos, S. Schaub-Meyer, C. Schroers, Neural inter-frame compression for video coding, in: Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 6421–6429.
- [36] E. Agustsson, D. Minnen, N. Johnston, J. Ballé, S.J. Hwang, G. Toderici, Scale-space flow for end-to-end optimized video compression, in: Proc. IEEE Conf. Comput. Vision Pattern Recog., 2020, pp. 8503–8512.
- [37] H. Liu, M. Lu, Z. Ma, F. Wang, Z. Xie, X. Cao, Y. Wang, Neural video coding using multiscale motion compensation and spatiotemporal context model, 2020, arXiv preprint arXiv:2007.04574.
- [38] M. Chen, A. Patney, A.C. Bovik, MOVI-codec: Deep video compression without motion, in: Proc. IEEE Picture Coding Symp., 2021, pp. 51–55.
- [39] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, in: Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 2017–2025.
- [40] A. Bruckstein, M. Elad, R. Kimmel, Down-scaling for better transform compression, IEEE Trans. Image Process. 12 (9) (2003) 1132–1144.
- [41] W. Lin, L. Dong, Adaptive downsampling to improve image compression at low bit rates, IEEE Trans. Image Process. 15 (9) (2006) 2513–2521.
- [42] X. Wu, X. Zhang, X. Wang, Low bit-rate image compression via adaptive down-sampling and constrained least squares upconversion, IEEE Trans. Image Process. 18 (3) (2009) 552–561.
- [43] M. Bhat, J.-M. Thiess, P.L. Callet, Can small be beautiful?: Assessing image resolution requirements for mobile TV, in: Proc. 13th Annu. ACM Int. Conf. Multimedia, 2005, pp. 829–838.
- [44] G. Cermak, M. Pinson, S. Wolf, The relationship among video quality, screen resolution, and bit rate, IEEE Trans. Broadcast. 57 (2) (2011) 258–262.
- [45] G. Georgis, G. Lentaris, D. Reisis, Reduced complexity superresolution for low-bitrate video compression, IEEE Trans. Circuits Syst. Video Technol. 26 (2) (2016) 332–345.
- [46] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, P. Frossard, Optimal selection of adaptive streaming representations, ACM Trans. Multimed. Comput. Commun. Appl. 11 (2s) (2015) 1–26, <http://dx.doi.org/10.1145/2700294>.
- [47] C. Li, L. Toni, P. Frossard, H. Xiong, J. Zou, Complexity constrained representation selection for dynamic adaptive streaming, in: Proc. IEEE Vis. Commun. Image Process., 2016.
- [48] Y. Sani, A. Mauthe, C. Edwards, Adaptive bitrate selection: A survey, IEEE Commun. Surv. Tutor 19 (4) (2017) 2985–3014.
- [49] M. Afonso, F. Zhang, D.R. Bull, Video compression based on spatio-temporal resolution adaptation, IEEE Trans. Circuits Syst. Video Technol. 29 (1) (2019) 275–280.
- [50] F. Zhang, M. Afonso, D.R. Bull, ViSTRA2: Video coding using spatial resolution and effective bit depth adaptation, Signal Process., Image Commun. 97 (2021) 116355, <http://dx.doi.org/10.1016/j.image.2021.116355>.
- [51] M. Bhat, J.-M. Thiess, P.L. Callet, A case study of machine learning classifiers for real-time adaptive resolution prediction in video coding, in: Proc. IEEE Int. Conf. Multimedia Expo, 2020, pp. 1–6.
- [52] M. Shen, P. Xue, C. Wang, Down-sampling based video coding using super-resolution technique, IEEE Trans. Circuits Syst. Video Technol. 21 (6) (2011) 755–765.
- [53] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, Y. Chen, Y. Wang, P. Wilkins, Y. Xu, J. Bankoski, A technical overview of AV1, Proc. the IEEE 109 (9) (2021) 1435–1462.
- [54] T.-S. Chang, Y.-C. Sun, L. Zhu, J. Lou, Adaptive resolution change for versatile video coding, in: Proc. IEEE Vis. Commun. Image Process., 2020.
- [55] T. Fu, K. Zhang, L. Zhang, S. Wang, S. Ma, An enhanced reference structure for reference picture resampling (RPR) in VVC, in: Proc. IEEE Int. Conf. Image Process., 2021.
- [56] M.A. Albreem, A.H.A. Habbash, A.M. Abu-Hudrouss, S.S. Ikki, Overview of precoding techniques for massive MIMO, IEEE Access 9 (2021) 60764–60801.
- [57] C. Dong, C.C. Loy, K. He, X. Tang, Image super-resolution using deep convolutional networks, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2016) 295–307.
- [58] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recog., 2016, pp. 770–778, <http://dx.doi.org/10.1109/cvpr.2016.90>.
- [59] Kodak lossless true color image suite, 2007, URL <https://r0k.us/graphics/kodak/>.
- [60] N. Asuni, A. Giachetti, TESTIMAGES: a large-scale archive for testing visual devices and basic image processing algorithms, in: Proc. Eurographics Italian Chapter Conference, 2014, pp. 63–70, <http://dx.doi.org/10.2312/stag.20141242>.
- [61] JPEG-AI call for evidence - IEEE MMSP2020 challenge (dataset), 2020, URL http://jpegai.github.io/test_images/.
- [62] Dataset of the CVPR workshop and Challenge on Learned Image Compression (CLIC), 2020, URL <http://clic.compression.cc/>.
- [63] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proc. Int. Conf. Learn. Represent., 2015, pp. 1–15.
- [64] G. Bjøntegaard, Calculation of average PSNR differences between RD-curves, 2001, Document VCEG-M33, ITU-T Video Coding Experts Group (VCEG) Thirteenth Meeting (Austin, TX, April 2001).
- [65] Z. Wang, E.P. Simoncelli, A.C. Bovik, Multi-scale structural similarity for image quality assessment, in: Proc. IEEE Asilomar Conf. on Signals, Syst., and Comput., 2003, pp. 1398–1402.
- [66] H.R. Sheikh, A.C. Bovik, Image information and visual quality, IEEE Trans. Image Process. 15 (2) (2006) 430–444, <http://dx.doi.org/10.1109/TIP.2005.859378>.
- [67] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, J.D. Cock, VMAF: The journey continues, 2018, NETFLIX Tech Blog, URL <https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12>.
- [68] J. Ascenso, P. Akayzi, M. Testolina, A. Boev, E. Alshina, Performance evaluation of learning based image coding solutions and quality metrics, 2019, Document ISO/IEC JTC 1/SC29/WG1 N85013, 85th JPEG Meeting.
- [69] M. Testolina, E. Upenik, J. Ascenso, F. Pereira, T. Ebrahimi, Performance evaluation of objective image quality metrics on conventional and learning-based compression artifacts, in: Proc. 13th Int. Conf. Quality Multimedia Exper., 2021.
- [70] R.A. Bradley, M.E. Terry, Rank analysis of incomplete block designs: The method of paired comparisons, Biometrika 39 (3–4) (1952) 324–345.
- [71] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: Proc. IEEE Conf. Comput. Vision Pattern Recog., 2018, pp. 586–595.
- [72] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612, <http://dx.doi.org/10.1109/tip.2003.819861>.
- [73] N. Ponomarenko, F. Silvestri, K. Egiazarian, M. Carli, J. Astola, V. Lukin, On between-coefficient contrast masking of DCT basis functions, in: 3rd Int. Workshop Video Process. Quality Metrics Consum. Electron, 2007.
- [74] L. Zhang, L. Zhang, X. Mou, D. Zhang, FSIM: A feature similarity index for image quality assessment, IEEE Trans. Image Process. (ISSN: 1057-7149) 20 (8) (2011) 2378–2386, <http://dx.doi.org/10.1109/TIP.2011.2109730>.
- [75] L. Zhang, Y. Shen, H. Li, VSI: A visual saliency-induced index for perceptual image quality assessment, IEEE Trans. Image Process. 23 (10) (2014) 4270–4281, <http://dx.doi.org/10.1109/tip.2014.2346028>.