CONVOLUTIONAL BLOCK DESIGN FOR LEARNED FRACTIONAL DOWNSAMPLING

Li-Heng Chen⋆   *Christos G. Bampis*†   *Zhi Li*†   *Chao Chen*†   *Alan C. Bovik*⋆

⋆ Laboratory for Image and Video Engineering (LIVE), The University of Texas at Austin
† Netflix, Inc.

ABSTRACT

The layers of convolutional neural networks (CNNs) can be used to alter the resolution of their inputs, but the scaling factors are limited to integer values. However, in many image and video processing applications, the ability to resize by a fractional factor would be advantageous. One example is conversion between resolutions standardized for video compression, such as from 1080p to 720p. To solve this problem, we propose an alternative building block, formulated as a conventional convolutional layer followed by a differentiable resizer. More concretely, the convolutional layer preserves the resolution of the input, while the resizing operation is fully handled by the resizer. In this way, any CNN architecture can be adapted for non-integer resizing. As an application, we replace the resizing convolutional layer of a modern deep downsampling model by the proposed building block, and apply it to an adaptive bitrate video streaming scenario. Our experimental results show that an improvement in coding efficiency over the conventional Lanczos algorithm is attained, in terms of PSNR, SSIM, and VMAF on test videos.

Index Terms— convolutional neural networks, downsampling, adaptive video streaming, perceptual video quality.

1. INTRODUCTION

Video signals have increasingly dominated the Internet in recent years, driven by the evolution of consumer electronics and the tremendous popularity of video sharing and streaming platforms. In a streaming video workflow, each component plays an important role in the end-to-end efficiency. For example, raw video sources directly determine the base video quality, while the selection of encoding parameters or rate control algorithms affect rate-distortion tradeoffs. An important recent advance is adaptive streaming framework, a technique that is widely used by video streaming services like Netflix, Youtube, or Facebook to improve the quality of experience of viewers [1–3].

As shown in the workflow illustration in Fig. 1, a source video segment (typically a scene) is downscaled into multiple resolutions using scaling factors $M \in \mathbb{Q}_+$. The videos at each resolution are then encoded using different quantization parameters (QPs), yielding a variety of rate-quality trade-
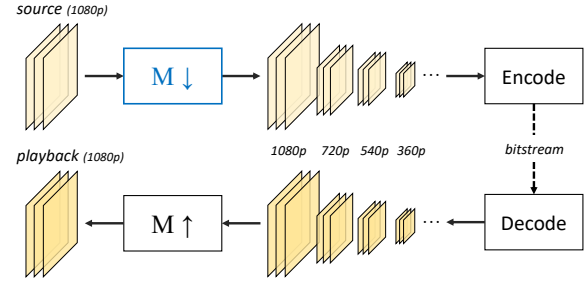

Fig. 1: A general flow diagram of adaptive video streaming. The parameter $M \geq 1$ denotes the current scaling factor, which may be varied. The downsampling block highlighted in blue is where our ideas are applied.

offs. From amongst the generated resolution-bitrate representations, a "best" video chunk is determined (typically by perceptual optimization), then streamed [4–6]. On the client side, the bitstream is decoded and scaled back to the device resolution prior to display. It is important to understand that the scaling factor $M \geq 1$ may be any reasonable rational number. For instance, streaming a 1080p source at 720p resolution ($M = 1.5$) is one of the most common choices for streaming services. The upscaling algorithms implemented on different display devices generally vary, and are not known by the stream providers. Hence, the encoding pipeline cannot be optimized for a specific upsampling algorithm.

Recently, deep neural networks have been applied to solve a wide diversity of video processing problems [7–10], including architectures designed for image resizing. Unlike legacy resizing algorithms, which rely heavily on conventional signal processing concepts, we optimize the parameters of a learned resizer in an end-to-end manner. In this way, we seek to improve adaptive streaming pipelines via CNN-based downsampling protocols (to implement the blue block in Fig. 1). From a practical perspective, several design facets need consideration:

1. Model complexity.
2. Agnostic to video codec and upsampling algorithm.
3. Allowing arbitrary (non-integer) scaling factors.

Item 3. is a very important, but often neglected, feature in the design of video resizers in compression workflows. However, currently available convolutional layers can only resize their inputs by integer factors. In order to address this problem, we
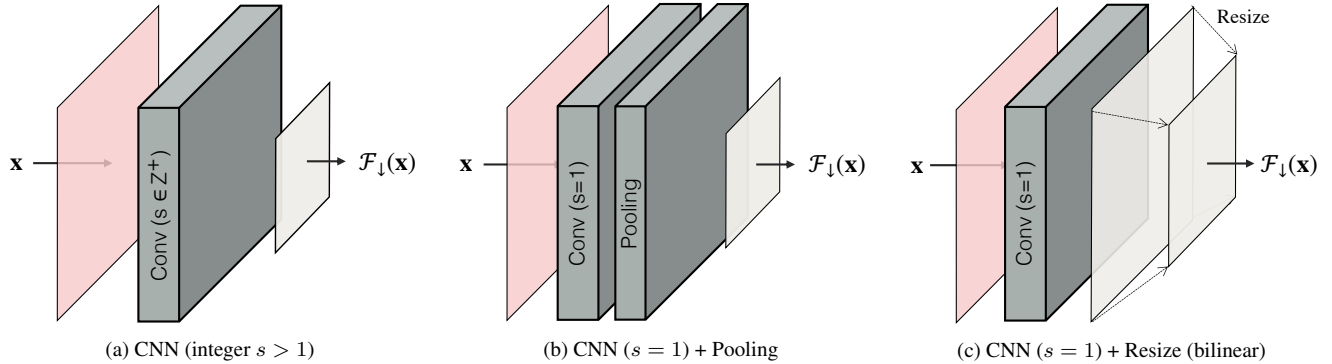
(a) CNN (integer $s > 1$)   (b) CNN ($s = 1$) + Pooling   (c) CNN ($s = 1$) + Resize (bilinear)

Fig. 2: Comparison of three convolutional blocks with down-sampled output. (a) A CNN block resizes by controlling the integer stride parameter s (integer resizing factor). (b) A CNN block with $s = 1$ resizes using an additional pooling layer (integer resizing factor) (c) Our proposed resizing module is constructed as a convolutional layer with $s = 1$ followed by a resize operation, allowing for arbitrary resizing factors.

propose a simple alternative block, which is constructed as a convolutional layer followed by a bilinear resizer.

The outline of this paper is as follows. Section 2 reviews related literature, while Section 3 presents details of the proposed convolutional block that allows non-integer resizing factors. Experiments and analysis are presented in Section 4, and finally, we conclude the paper in Section 5.

2. RELATED WORK

Resizing Algorithms. Beyond early approaches, such as bilinear, bicubic, and Lanczos, a variety of models for image and video scaling have been proposed. More recently, patch-based methods have been proposed that exploit intra [11–13] or inter [14–16] similarities, while CNN-based models [17–21] have produced excellent results. However, these kinds of upscaling models are implemented on the device side, making them not inherently controllable components to the encoding side. Thus, they are of less interest in our context, since, their usability is limited in streaming workflows.

Optimized approaches to resolution reduction include better aligning local image features [22], or preserving perceptually important details [23–25]. More recently, deep learning based downsampling models have included CNN-CR [26], which applies a 10-layer CNN to learn residuals on top of bicubic downsampled images, and a content adaptive resampler (CAR) model which estimates resampling kernels [27].

Video Quality Assessment. Another topic relevant to adaptive video streaming is the prediction of *perceptual* quality. PSNR has been shown to be inconsistent to human perception [28], especially in measuring specific distortions, such as scaling artifacts. Fortunately, many video quality models [29–34] have been proposed in recent years. In particular, SSIM [35] and VMAF [36] have been very widely deployed to optimize a large fraction of compressed Internet video traffic, including downsampling conducted as part of compres-

sion.

3. LEARNING A DOWNSAMPLER FOR ADAPTIVE VIDEO STREAMING

3.1. Proposed Convolutional Block

We begin by comparing two commonly used convolutional blocks, shown in Figs. 2(a) and 2(b). Given an input $\mathbf{x}$ with spatial resolution $\text{W} \times \text{H}$, a convolutional block has trainable hyper parameters outputs $\mathcal{F}_{\downarrow M}(\mathbf{x})$ yielding a reduced resolution $\text{W}/M \times \text{H}/M$. Normally, the downsample operation by a factor M is done by (Fig. 2(a))

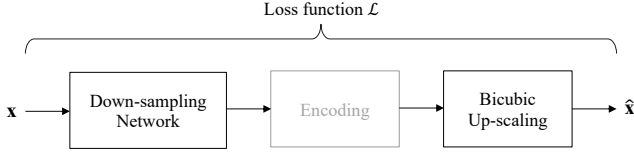$$\mathcal{F}_{\downarrow M}(\mathbf{x}) = \mathcal{C}_{s=M}(\mathbf{x}), \tag{1}$$

where $\mathcal{C}_s$ denotes a convolutional operation with a `stride` parameter s. Alternatively, pooling layer $\mathcal{P}_{\downarrow M}$, typically using max pooling or average pooling, is used to reduce resolution (Fig. 2(b))

$$\mathcal{F}_{\downarrow M}(\mathbf{x}) = \mathcal{P}_{\downarrow M}\left[\mathcal{C}_{s=1}(\mathbf{x})\right]. \tag{2}$$
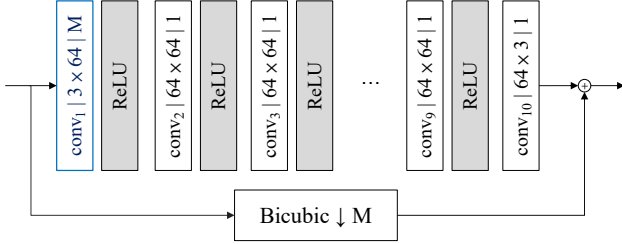
Unfortunately, the two aforementioned blocks only allow for *integer* scaling factors $M \in \mathbb{Z}_+$, which may limit needed flexibility when implementing resolution changes in broader applications. To address this problem, we instead replace $\mathcal{P}$ in (2) by a differentiable resizer $\mathcal{R}$ that supports arbitrary scaling factors $M \in \mathbb{Q}_+$ (Fig. 2(c))

$$\mathcal{F}_{\downarrow M}(\mathbf{x}) = \mathcal{R}_{\downarrow M}\left[\mathcal{C}_{s=1}(\mathbf{x})\right], \tag{3}$$

which we dub the `conv-resize` block. We realize the proposed building block as a convolutional layer `conv` followed by a `bilinear` resizer. Of course, the entire network can be trained end-to-end by back-propagating through the forward model, and which can be easily implemented using common libraries (e.g., TensorFlow or PyTorch). Interestingly, a similar "`resize-conv`" block has been used to mitigate artifacts arised from uneven overlapped responses in transposed

(a) Training framework. The gray block is not present in training.

(b) Network architecture of CNN-CR [26].

Fig. 3: Detailed (a) training framework of a downsampling network for adaptive video streaming, and (b) CNN-CR network architecture. The convolutional parameters are denoted as: input channel × output channel | stride. Here, we replace the first convolutional layer, highlighted in blue, by a `conv-resize` layer.

convolution [37]. However, this concept has not been extended to solve the problem we discuss here.

3.2. End-to-End Training of Downsampling Network

Training Framework. Aiming to end-to-end optimize the downsampling network for easy insertion into adaptive streaming scenarios, we constructed a training framework similar to the encoding pipeline in Fig. 1. As shown in Fig. 3(a), the input training data is down-scaled by a network with trainable parameters, encoded, and reconstructed to the original resolution by an upscaler. Since all conventional hybrid video encoders, such as H.264, are not *differentiable* components, they are not feasible for back-propagation. Thus, we relax this problem by simply removing the encoder from the pipeline. We implemented the upscaling algorithm using bicubic interpolation as a generic and reasonably high-performance comparison. As we will show, this design achieves consistent performance gain for all the widely adopted upscaling algorithms.

Given an input source batch $\mathbf{x}$ and a reconstructed batch $\hat{\mathbf{x}}$ presented in RGB color space, the loss function is defined as the residual between $\mathbf{x}$ and $\hat{\mathbf{x}}$ of a distance function d:

$$\mathcal{L} = d(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2. \tag{4}$$

Here, we use the mean squared error loss ($d(x) = \|x\|_2^2$), which maximizes the PSNR of the reconstructed images.

Network Architecture. We use the previously mentioned CNN-CR model as the backbone of the downsampling network. As shown in Fig. 3(b), the 3-channel RGB signal is fed into the network. The architecture of CNN-CR consists of 10 stages of convolutional layers. The sizes of the convolution

kernels are fixed at 3×3 in all the layers, while the number of filters is 64 for all of the first 9 stages. Except for the conv_{10} layer, all of the convolutional layers are activated by a ReLU nonlinearity. Finally, a 3-channel output is produced having reduced size, yielding a residual added to the bicubic down-sampled input image. The parameterization of each layer is detailed in the figure. It should be note that the sub-sampling process happens in the first convolution layer, by the stride $s = M$. To remove the limitation of an integer M, the first layer conv_1 is replaced by our proposed `conv-resize` block.

4. EXPERIMENTS

4.1. Implementation Details and Experimental Setup

We used the TensorFlow framework (version 1.15) to implement the proposed non-integer deep downsampling method. The Adam solver [38] was used to optimize the networks, with parameters $(\beta_1, \beta_2) = (0.9, 0.999)$ and a batch size of 16. The networks were trained on 500K iterations of back-propagation, with a learning rate that was fixed at $1e - 4$. The training images were randomly cropped to $M \lfloor \frac{256}{M} \rfloor \times M \lfloor \frac{256}{M} \rfloor$, which is divisible by the scaling factor M.

We used DIV2K [39], an image dataset consisting of 1000 very high quality pictures,[1] as training data. This dataset was designed for studying image super-resolution problems. All of the images in it have 2K pixels along either the vertical or horizontal axis. To evaluate our method under the adaptive video streaming scenario, we utilized 20 test video contents of 1080p resolution and YUV420 format obtained from Xiph Video Test Media,[2] We also used 25 1080p video sources collected from the Netflix library, yielding more diverse and realistic contents. To integrate the network into a video encoding pipeline, the video formats input to the network were in RGB888 format. Following resolution reduction, the videos were converted back to their original format (YUV420) prior to encoding.

4.2. Quantitative Comparison

We measured the objective coding efficiency of each down-sampling model within the same video encoding pipeline using the Bjøntegaard-Delta bitrate (BD-rate) [40], which quantifies average differences in bitrate at the same distortion level relative to another reference encode. To calculate BD-rate, we encoded the down-sampled videos by x264 at 15 different QPs, ranging from 17 to 46. Then, the encoded videos were up-scaled back to their original resolutions. The performances of all of the downsamplers were compared to the same baseline - the Lanczos downsampling algorithm implemented in ffmpeg. A negative number of BD-rate means the

---

[1] https://data.vision.ee.ethz.ch/cvl/DIV2K/
[2] https://media.xiph.org/video/derf/

Upsampler	bilinear↑						bicubic↑					
Downsampler	CNN-CR↓			Proposed↓			CNN-CR↓			Proposed↓		
BD-rate metric	PSNR	SSIM	VMAF	PSNR	SSIM	VMAF	PSNR	SSIM	VMAF	PSNR	SSIM	VMAF
$M = 1.5$	—	—	—	-4.06	-2.47	-1.20	—	—	—	-2.22	-1.19	-0.77
$M = 2$	-4.84	-4.90	-2.45	-4.72	-4.74	-2.28	-2.68	-3.18	-2.16	-2.64	-3.11	-2.11
$M = 2.5$	—	—	—	-4.59	-6.54	-6.88	—	—	—	-2.56	-4.53	-5.34
$M = 3$	-4.27	-8.00	-10.51	-4.51	-8.17	-11.26	-2.86	-6.33	-8.78	-2.77	-6.55	-9.25
$M = 4$	-4.06	-10.36	-14.70	-5.48	-11.88	-15.97	-1.31	-7.42	-12.99	-3.11	-9.14	-15.12
$M = 5$	-2.01	-9.72	-22.37	-3.28	-11.37	-24.17	+0.52	-6.62	-19.24	+0.16	-7.74	-20.20

Table 1: Comparison of the performances of a conventional convolutional layer and the proposed `conv-resize` layer deployed in CNN-CR: average change of BD-rate expressed as percentage. The baseline of comparison is the Lanczos downsampling algorithm with the same encoding recipe. Smaller or negative values indicate better coding efficiency. A "—" in a cell indicates the result is not applicable to the model.

Scaling factor M	1.5	2	2.5	3	4	5
`conv-resize`	-2.22	-2.64	-2.56	-2.77	-3.11	+0.16
`resize-conv`	-0.98	-1.46	+6.73	+17.27	+18.88	+31.51

Table 2: Comparison of using `conv-resize` and `resize-conv` as the first layer of CNN-CR↓ with bicubic↑. Each cell presents the average PSNR BD-rate expressed as percentage.

bitrate was reduced as compared with the baseline. Lastly, the distortion levels that were used for BD-rate calculation were quantified using PSNR, SSIM, and VMAF.

The performance results are shown in Table 1, with respect to different objective video quality models. We report the BD-rate changes obtained relative to the baseline (Lanczos downsampling under the same conditions), averaged over all the videos in the test set. We comprehensively evaluated the proposed downsampling networks for various scaling factors that are commonly used in practice, using two different upsampling algorithms. From the results in the table, we can draw a number of conclusions. First, these results show that learned downsampling models are able to further optimize existing video encoding pipelines. Indeed, significant BD-rate reductions were obtained in many cases. The proposed `conv-resize` block performed closely to the conventional block at integer scaling factors, and yielded reasonable BD-rates when M was not an integer. It may also be observed that, when using bicubic upsampling when $M = 5$, the PSNR BD-rate was slightly worse than the baseline scenario. It is possible that the models were trained without considering encoding effects, such as distortions and rate consumption, hence resulting in a suboptimal result. However, the subtle suboptimality in PSNR BD-rate is negligible, since PSNR is not the ultimate optimization target.

Despite training with a fixed bicubic upsampler, the models were still able to generalize well to bilinear upsampling. In fact, the performance results reveal similar trends on the two upsampling algorithms. However, more significant BD-rate improvements were obtained on bilinear upsampling, which usually results in worse quality, leaving greater room

for improvement of objective video quality. We have also observed very similar results using other upscaling algorithms, including Lanczos. Another interesting observation that can be made is that the CNN-based models delivered coding gains with respect to all of the BD-rate measurements.

4.3. Which Goes First? Convolution or Resizing?

By combining the downsampling process with a resizer, non-integer scaling is preserved if the order of `conv` and `resize` is reversed. To study this alternate model, we compared two blocks using both orderings in Table 2. It may be observed that, on all the scaling factors, the proposed block `conv-resize` delivered significantly better results than `resize-conv` on the task of video downsampling. This is not surprising, since the interpolation video signals were decimated, with loss of information, at the input of the network. Placing a learnable convolutional layer beforehand allows the retention of information, yielding better performance. This is quite analogous to the classic signal processing rate change structure of an anti-aliasing filter placed prior to a decimator. We also observed that the performance of `resize-conv` dropped sharply with increases of M, as a consequence of more severe information loss.

5. CONCLUSION

We designed a convolutional block that allows fractional resizing in downsampling networks. The proposed block is simple and can be effectively implemented within diverse deep learning platforms. We believe that the new block can also be applied in different types of downsampling architectures. Looking further ahead, we plan to extend the idea to other important problems, such as the quality assessment of pictures of different scales, or viewed at different distances.

6. REFERENCES

[1] C. Chen, Y.-C. Lin, S. Benting, and A. Kokaram, "Optimized transcoding for large scale adaptive streaming using playback statistics," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2018, pp. 3269–3273.

[2] I. Katsavounidis, "Dynamic optimizer – a perceptual video encoding optimization framework," *The NETFLIX tech blog*, 2018. [Online]. Available: https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f

[3] P.-H. Wu, V. Kondratenko, and I. Katsavounidis, "Fast encoding parameter selection for convex hull video encoding," in *Proc. SPIE Applications Digital Image Process. XLIII*, Aug. 2020.

[4] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard, "Optimal selection of adaptive streaming representations," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 11, no. 2s, pp. 1–26, Feb. 2015.

[5] C. Li, L. Toni, P. Frossard, H. Xiong, and J. Zou, "Complexity constrained representation selection for dynamic adaptive streaming," in *Proc. IEEE Visual Commun. Image Process.*, Nov. 2016.

[6] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive bitrate selection: A survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2985–3014, 2017.

[7] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Jun. 2012, pp. 2392–2399.

[8] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–27.

[9] Y.-L. Liu, Y.-T. Liao, Y.-Y. Lin, and Y.-Y. Chuang, "Deep video frame interpolation using cyclic frame generation," in *Proc. AAAI*, 2019, pp. 8794–8802.

[10] S. Paul, A. Norkin, and A. C. Bovik, "Speeding up VP9 intra encoder with hierarchical deep learning-based partition prediction," *IEEE Trans. Image Processing*, vol. 29, pp. 8134–8148, 2020.

[11] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vision*, Sep. 2009.

[12] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. Graph.*, vol. 30, no. 2, pp. 1–11, Apr. 2011.

[13] A. Singh and N. Ahuja, "Super-resolution using sub-band self-similarity," in *Proc. Asia Conf. Comput. Vision*, 2015, pp. 552–568.

[14] W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, 2002.

[15] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Jun. 2004, pp. 2392–2399.

[16] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 6, pp. 1127–1133, Jun. 2010.

[17] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[18] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proc. IEEE Int. Conf. Comput. Vision*, Dec. 2015.

[19] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Jun. 2016, pp. 1646–1654.

[20] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Jul. 2017, pp. 3147–3155.

[21] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2020.

[22] J. Kopf, A. Shamir, and P. Peers, "Content-adaptive image downscaling," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–8, Nov. 2013.

[23] A. C. Öztireli and M. Gross, "Perceptually based downscaling of images," *ACM Trans. Graphics*, vol. 34, no. 4, pp. 1–10, Jul. 2015.

[24] N. Weber, M. Waechter, S. C. Amend, S. Guthe, and M. Goesele, "Rapid, detail-preserving image downscaling," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–6, Nov. 2016.

[25] J. Liu, S. He, and R. W. H. Lau, "L_0-regularized image downscaling," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1076–1085, Mar. 2018.

[26] Y. Li, D. Liu, H. Li, L. Li, Z. Li, and F. Wu, "Learning a convolutional neural network for image compact-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1092–1107, Mar. 2019.

[27] W. Sun and Z. Chen, "Learned image downscaling for upscaling using content adaptive resampler," *IEEE Trans. Image Process.*, vol. 29, pp. 4027–4040, 2020.

[28] Z. Wang and A. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

[29] K. Seshadrinathan and A. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 335–350, Feb. 2010.

[30] P. V. Vu, C. T. Vu, and D. M. Chandler, "A spatiotemporal most-apparent-distortion model for video quality assessment," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 2505–2508.

[31] M. H. Pinson, L. K. Choi, and A. C. Bovik, "Temporal video quality model accounting for variable frame delay distortions," *IEEE Trans. Broadcast.*, vol. 60, no. 4, pp. 637–649, Dec. 2014.

[32] C. G. Bampis, P. Gupta, R. Soundararajan, and A. C. Bovik, "SpEED-QA: Spatial efficient entropic differencing for image and video quality," *IEEE Signal Process. Lett.*, vol. 24, no. 9, pp. 1333–1337, Sep. 2017.

[33] Z. Tu, J. Lin, Y. Wang, B. Adsumilli, and A. C. Bovik, "BBAND index: A no-reference banding artifact predictor," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process. (ICASSP)*, 2020, pp. 2712–2716.

[34] Z. Tu, Y. Wang, N. Birkbeck, B. Adsumilli, and A. C. Bovik, "UGC-VQA: Benchmarking blind video quality assessment for user generated content," *arXiv preprint arXiv:2005.14354*, 2020.

[35] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Process. Image Commun.*, vol. 19, no. 2, pp. 121–132, Feb. 2004.

[36] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The NETFLIX Tech Blog*, 2016. [Online]. Available: https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652

[37] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: https://distill.pub/2016/deconv-checkerboard/

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[39] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vision Pattern Recog. Workshops*. IEEE, Jul. 2017, pp. 1122–1131.

[40] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," *document VCEG-M33, ITU-T Video Coding Experts Group (VCEG) Thirteenth Meeting*, Austin, TX, April 2001.