Adaptation and Attention for Neural Video Coding

Nannan Zou[1, 2], Honglei Zhang[1], Francesco Cricri[1], Ramin G. Youvalari[1], Hamed R. Tavakoli[1]
Jani Lainema[1], Emre Aksu[1], Miska Hannuksela[1], Esa Rahtu[2]
[1]Nokia Technologies, [2]Tampere University, Tampere, Finland
nannan.zou@nokia.com

***Abstract*—Neural image coding represents now the state-of-the-art image compression approach. However, a lot of work is still to be done in the video domain. In this work, we propose an end-to-end learned video codec that introduces several architectural novelties as well as training novelties, revolving around the concepts of adaptation and attention. Our codec is organized as an intra-frame codec paired with an inter-frame codec. As one architectural novelty, we propose to train the inter-frame codec model to adapt the motion estimation process based on the resolution of the input video. A second architectural novelty is a new neural block that combines concepts from split-attention based neural networks and from DenseNets. Finally, we propose to overfit a set of decoder-side multiplicative parameters at inference time. Through ablation studies and comparisons to prior art, we show the benefits of our proposed techniques in terms of coding gains. We compare our codec to VVC/H.266 and RLVC, which represent the state-of-the-art traditional and end-to-end learned codecs, respectively, and to the top performing end-to-end learned approach in 2021 CLIC competition, `E2E_T_OL`. Our codec clearly outperforms `E2E_T_OL`, and compare favorably to VVC and RLVC in some settings.**

***Index Terms*—learned video codec, split attention, content-adaptive, overfitting, finetuning**

I. Introduction

Nowadays, image codecs based on deep learning represent the state-of-the-art when considering MS-SSIM [1] and PSNR [2] visual quality metrics. The typical architecture is based on the auto-encoder, where the encoder and decoder neural networks perform non-linear forward and inverse transform, respectively. The output of the encoder is typically lossless encoded by an arithmetic encoder, using a learned probability model. In the video domain, however, the state-of-the-art is represented by traditional codecs such as VVC/H.266 [3] and HEVC/H.265 [4] standards, which follow the prediction-transform paradigm: intra-frame and inter-frame prediction, followed by transform-coding of prediction residuals. We propose a learned codec that follows a common design [5], [6] inspired by traditional codecs, where a learned image codec performs intra-frame coding, and a conditional interpolation model interpolates the other frames based on reconstructed intra frames. Based on the observation that the extent by which objects move in terms of pixels depends also on the spatial resolution, in our inter-frame codec, the output of motion estimation is adapted by the input video's resolution. Recently, efficient attention blocks have been proposed, such as the Split Attention block (or ResNeSt block) [7], which applies the squeeze-and-attention concept [8] [9] to groups of feature maps. ResNeSt blocks have already been successfully used in [2] as one of the blocks within a learned image codec. We further extend the ResNeSt block idea by designing the Dense Split Attention (DSA) block, that combines the efficiency of split attention with the power of dense connections [10] between each of a set of ResNet blocks and the output of split attention. To further optimize part of the codec to the input content at inference time, several prior works have proposed to optimize or *overfit* some of the encoder's parameters [11] [12] [13], or the latent tensor output by the encoder [14] [15], or some or all the decoder's parameters [16] [17], or both latent tensor and decoder's parameters [18]. Overfitting decoder's parameters would incur into a bitrate overhead for providing the weight-update to the decoder side. To limit such overhead in the case of a VVC codec augmented with a decoder-side post-processing neural network, the authors of [17] propose to overfit only the bias terms of all the convolutional layers on all the frames in a Group of Pictures (GOP). Our overfitting process is applied on multiplicative parameters instead of additive bias terms. Also, we argue that *(i)* the layers of a neural network are not equally important, thus we overfit only a selected subset of parameters, and *(ii)* overfitting on a single frame is enough for short sequences without scene changes.

In summary, we propose a highly-adaptive neural video codec, which includes the following contributions: *(i)* resolution-adaptive motion estimation, *(ii)* Dense Split Attention blocks, *(iii)* overfitting of multiplicative parameters on a single frame per video, *(iv)* adapting the combination of forward and backward predictions on each frame.

II. Proposed Methods

Our video codec is organized as an intra-frame codec and an inter-frame codec. The intra-frame codec processes one frame every eight frames, without using any information from other frames. The seven frames between two consecutive intra-frames are coded by the inter-frame codec in a hierarchical sequential manner. I.e., first the frame with index 4 (relative to the start of the intra-frame period) is coded based on intra-coded frames $\{0, 8\}$, then frame 2 and frame 6 are coded based on intra or inter-coded frames $\{0, 4\}$ and $\{4, 8\}$, respectively, finally frames 1, 3, 5 and 7 are coded based on intra or inter-coded frames $\{0, 2\}$, $\{2, 4\}$, $\{4, 6\}$ and $\{6, 8\}$, respectively.

A. Intra-Frame Codec

Our intra-frame codec, shown in Figure 1, has an autoencoder architecture similar to other end-to-end learned image codecs [1], [19], [20]. The encoder transforms the input frame

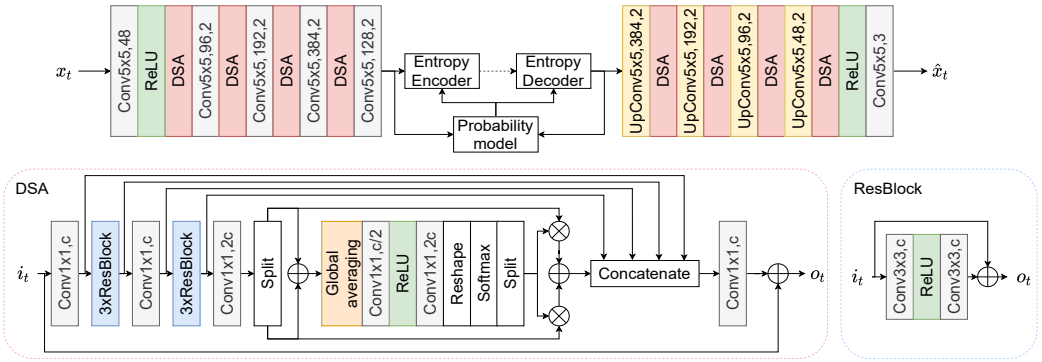

Fig. 1: Our intra-frame codec, with the proposed Dense Split Attention (DSA) block. "Conv$K\times K$,c,s" and "UpConv$K\times K$,c,s" stand for 2D convolutional layer and 2D transposed convolutional layer, respectively, with kernel size $K\times K$, c output channels, and an optional stride value s if it is not 1.

to a latent representation, which is quantized using uniform scalar quantization (the quantizer is not shown for simplicity). The Probability Model component estimates the probability of each element in the latent representation, and provides it to the entropy codec, which is an arithmetic codec. The decoder has a mirrored architecture with respect to the encoder.

One novel component of our intra-frame codec is the Dense Split Attention (DSA) block, which is shown in detail in Figure 1. This block combines the efficiency of split channel attention [7] with the power of dense connections [10]. In particular, DSA consists of an initial set of convolutional layers and ResBlock layers, followed by a core attention block whose output is concatenated with dense connections coming from the initial set of layers, before being processed by a final convolutional layer. The core attention block can be any attention-based block, such as non-local attention [19]. However, in order to keep computational and memory consumption low, we opted for the efficient Split Attention block [7] with $k = 1$ group and $r = 2$ splits of features.

Another novel aspect of our intra-frame codec is the Overfittable Multiplicative Parameters (OMPs). An OMP is a learnable parameter that multiplies a feature map output by a certain kernel of a convolutional layer. The OMPs are initialized to 1 and then optimized at inference time. However, we chose to use OMPs only within the last DSA block of the intra-decoder, more specifically on the output of the following layers: the first convolutional layer, the first convolutional layer of the first ResBlock, the second convolutional layer of the second and fourth ResBlock. These layers were selected empirically based on an evaluation on validation data.

B. Inter-Frame Codec

An overview of the inter-frame codec is provided in Figure 2. The inputs to the encoder are two reconstructed reference frames $\hat{x}_{t-d}$, $\hat{x}_{t+d}$ and the target frame x_t. In practice, we define d as the distance from the target frame x_t, which is randomly chosen among $\{1, 2, 4\}$. A reconstructed reference frame may be an intra-coded frame or a inter-coded frame. First, the Encoder Feature Pyramid Net extracts multi-scale features from the input frames. Next, the multi-scale features are aggregated by the Feature Encoder Net and transformed to a latent representation of the target frame. Then, the Entropy Encoder, in this case an arithmetic encoder, compresses the latent representation into a bitstream by using the output of the Probability Model. Our probability model is conditioned on features extracted from the reconstructed reference frames, by using the Entropy Feature Pyramid Net. The decoder takes the compressed bitstream of the latent representation as its input. The bitstream is first decompressed and dequantized by the Entropy Decoder to generate a reconstruction of the latent representation. The latent representation implicitly embeds forward motion information (from $\hat{x}_{t-d}$ to $\hat{x}_t$), backward motion information (from $\hat{x}_{t+d}$ to $\hat{x}_t$), and information about a residual signal. The reconstructed latent tensor is passed to the Feature Decoder Net that generates multi-scale motion features and a residual signal e_t. The Decoder Feature Pyramid Net extracts multi-scale features from reference frames. The Motion Estimation module takes in the multi-scale motion features and reference frame's multi-scale features to output motion information. The Motion Estimation has a similar architecture as FlowNet [6], [21]. Different from other systems [22], the Motion Estimation is randomly initialized and end-to-end learned together with other components in our system. The resolution information, represented by the height h and width w of the video frames, is embedded into feature space by the Resolution Embedding module. The embedded resolution is used to scale the motion information. The scaled motion information is then used by the Frame Prediction module to warp the reference frames into a forward prediction $\hat{f}_t^{fwd}$ and a backward prediction $\hat{f}_t^{bwd}$ of the target frame. These predictions are then combined with the residual and with the two closest intra-coded frames by the Combiner component to produce the final reconstructed target frame $\hat{x}_t$. We leave out details about several of the above modules because of space limitations. They are based on known neural network architectures and do not include major novelties. The Resolution Embedding module consists of two fully-connected layers followed by a Leaky ReLU layer. Regarding the Combiner, the

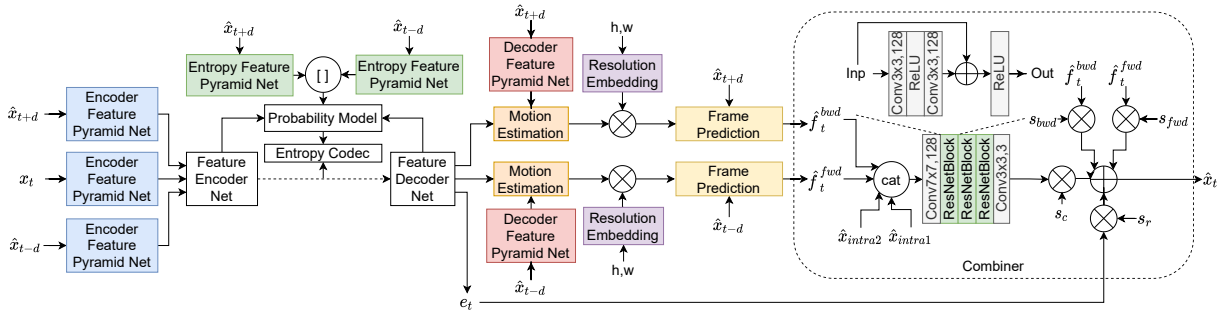

Fig. 2: Architecture of the inter-frame codec. In this figure, components that share weights are shown with the same color.

input forward and backward predictions of the target frame $\hat{f}_t^{fwd}$ and $\hat{f}_t^{bwd}$ are concatenated with the closest intra-coded frames $\hat{x}_{intra1}$ and $\hat{x}_{intra2}$, and then processed by a sequence of convolutional layers and ResNet blocks [23], obtaining a temporary prediction $\tilde{x}_t$. Then, $\tilde{x}_t$, $\hat{f}_t^{fwd}$, $\hat{f}_t^{bwd}$ and the decoded residual e_t are combined by a linear combination $\hat{x}_t = s_t^{tmp}\tilde{x}_t + s_t^{bwd}\hat{f}_t^{bwd} + s_t^{fwd}\hat{f}_t^{fwd} + s_t^e e_t$, where s_t^{tmp}, s_t^{bwd}, s_t^{fwd}, s_t^e are scalars that are trained during the training stage of the inter-frame codec, and overfitted at inference time.

C. Probability Model

Our probability models used in the intra-frame and inter-frame codec are based on the state-of-the-art probability model for lossless image compression described in [24]. An input latent representation is first downscaled to multiple resolutions. The representation in the lowest resolution is compressed/decompressed using a non-conditional distribution model with the assumption that the elements are independent and identically distributed. The distributions of the representations in other resolutions are conditioned on the representation in lower resolutions. At the encoding and decoding stage, the system first processes the representation in the lowest resolution. Then, the system moves to the next higher-resolution representation, using the low-resolution representation as a context to derive the distribution function. This procedure continues until the original latent representation is processed. To further improve the performance, we partition the elements in each representation into several groups. The elements in one group are processed together using the elements that have been processed as a context. We adopt the conditional Gaussian distribution model as used in [25] for the sake of accuracy and computational efficiency. The parameters of the distribution model, i.e., means and scales, are estimated using a deep neural network given the input of the low-resolution representation and the elements in the same resolution that have been processed. The probability model used in the inter-frame codec is enhanced by taking features extracted from the reconstructed reference frames as extra context for representations in resolutions other than the lowest one.

D. Training and Inference Aspects

Both the intra-frame codec and the inter-frame codec were trained on 256x256 patches, by using MS-SSIM and rate loss as the training objectives: $\mathcal{L} = \mathcal{D} + \lambda\mathcal{R}$, where the distortion $\mathcal{D}$ is the negative MS-SSIM, $\mathcal{R}$ is the rate derived from the probability model, and λ is a hyper-parameter. The intra-frame codec was trained only as a stand-alone module. The inter-frame codec was first pretrained as a stand-alone module, by using uncompressed reference frames, and then finetuned by using a similar pipeline as at inference time, i.e., by using intra-coded frames and inter-coded frames as reference frames in hierarchical sequential processing.

At inference time and for each intra frame, the latent tensor output by the intra-frame encoder is overfitted as in [15]. After that, the OMPs of the intra-frame decoder are optimized on the first intra frame of each video, and used for all intra frames of that video. This strategy leverages the high temporal redundancy of videos when there is no scene change. A strategy that overfits the OMPs jointly on all intra frames of a video would be more time-consuming, and the coding gains may not be worth the extra time. Another alternative strategy would overfit a separate set of OMPs for each intra frame. However, coding gains would be negatively affected by a much higher bitrate overhead required by the overfitted parameters. In the experimental section, we provide comparisons for some of these strategies. Based on our comparisons, we choose the first optimization strategy mentioned above for our intra codec decoder. After overfitting, the overfitted OMPs are uniformly quantized to 8 bits. If the quantized OMPs provide coding gains in terms of the overall loss over a video, the parameters would work as separate bitstreams together with latent tensor bitstreams for decoding. Otherwise, we do not include them into the bitstreams. We also propose to overfit, at inference time, the scaling parameters used within the Combiner module of the inter-frame decoder. These parameters are adapted separately on each inter-coded frame.

III. Experiments

The codec was trained on the CLIC 2021 video dataset. The intra-frame codec was trained by using all the frames of all the training videos for 60 epochs, with a learning rate of $5e{-}5$ and batch-size of 60 frames. The inter-frame codec was first pretrained on uncompressed reference frames for which the distance from the target frame was randomly chosen among $\{1, 2, 4\}$. This pretraining was performed for 34 epochs, a learning rate of $5e{-}5$ and a batch-size of 63

samples, where each sample consists of two reference frames and one target frame. The inter-frame codec is then finetuned on all frames of all videos in the dataset, for 10 epochs, with a learning rate of $2e{-}5$ and batch-size of 56 sets of 7 inter frames. We follow the evaluation framework of the CVPR workshop and challenge CLIC (video track), which is the most recent learned video coding conference, to allow for an easier comparison with prior art learned codecs. According to this framework, the combined size of the decoder and bitstreams, calculated as $decodersize + bitstreams/0.019$, should not exceed 1309MB. We tested our codec on the CLIC test set and on the JVET-CTC sequences. For JVET-CTC sequences, we excluded Class A due to the high resolution causing high memory consumption, and we converted 10 bits sequences to 8 bits for simplicity. We compared our codec to the state-of-the-art traditional and learned video codec, i.e., VVC/H.266 and RLVC [26], respectively.

For VVC, the VTM-12.0 software was used in our comparison. We tuned its hyper-parameters to achieve the target combined size on the CLIC dataset as close as possible. We evaluated it with an intra period of 8 frames (same as our codec) on CLIC test set, and both an intra period of 8 frames and an intra period of 1 second (default setting) on JVET-CTC sequences. To evaluate RLVC, the bi-IPPP GOP structure with default settings was adopted in our experiments. Additionally, we used their MS-SSIM model with lambda value of 8 which provides the smallest bitrate. Nonetheless, RLVC still cannot achieve the target combined size, as showed in Table I. As RLVC was trained on RGB data, and the test datasets are in YUV 4:2:0 color format, we converted the videos to RGB by using FFmpeg. We measured the quality drop caused by the conversion as the MS-SSIM and the peak signal-to-noise ratio (PSNR) computed on the original YUV data and corresponding YUV data obtained after converting to RGB and back to YUV. For CLIC test set, the MS-SSIM was 0.994, and the PSNR was 55.9 dB. For JVET-CTC sequences, the MS-SSIM was 0.999, and the PSNR was 50.5 dB. We provide test results both in RGB and YUV domains.

Table I reports the results for the above experiments. NNVC refers to the proposed codec. VVC8 is VVC with intra period of 8 frames. RLVC-RGB and RLVC-YUV are RLVC evaluated in RGB and YUV domain, respectively. We also report the performance of the top performing end-to-end learned approach in 2021 CLIC competition, `E2E_T_OL`. NNVC surpasses their performance. We measure the speed of NNVC on one NVIDIA Tesla V100 SXM2 GPU. For an 1280x720 CLIC video, our encoding (including the overfitting) and decoding run on average at 0.006 and 1.2 frames/sec, respectively. For the same video, VVC encodes and decodes on average at 0.022 and 23.6 frames/sec, respectively, with one Intel Xeon Gold 6154 CPU.

Regarding the intra-frame codec, we compare our DSA block with the "group-separated attention (GSA)" block used in the prior art learned image codec [2]. To this end, we designed a codec which is as similar as possible to our proposed codec, but which uses the GSA block. While DSA's ResBlock uses two convolutional layers, in [2] the residual block uses three convolutional layers. In our study, we compared both of these two cases. Table II describes the results of comparison. The "Score" values are derived as the negative of the loss values computed on the evaluation set by using $\lambda = 0.1$. A higher score value indicates better performance. DSA-N and GSA-N refer to the proposed NNVC's intra codec and the NNVC's intra codec where DSA blocks were replaced with GSA blocks [2], respectively. N is the number of convolutional layers in the ResBlock of DSA or GSA. The models were trained on 30 videos of the CLIC training set, and evaluated on 10% of the frames in those videos (similar setup as in CLIC competition). The results clearly show that DSA block performs better than GSA, even when GSA uses more layers and parameters (i.e., DSA-2 vs GSA-3).

In another experiment (see **Intra codec only** part in Table III), we compared different strategies of overfitting the OMPs. For simplicity, we trained on a subset of the CLIC training set, and evaluated on the CLIC validation set. In one strategy (c_1), that we adopted in our final codec, we overfit a few selected layers of last DSA block in the intra-frame decoder by using the first intra frame of each video, and then employ the OMPs for all intra frames of that video; in another strategy (c_2), we overfit the OMPs of all layers of last DSA block, and the very last layer of the decoder, separately on each intra frame. The Score was computed by using $\lambda = 0.15$. In the strategy c_2, the overfitted OMPs still required a much higher bitrate overhead after quantization. Since we include the quantized OMPs into bitstream only if the parameters provide coding gains in terms of the overall loss (over a video for c_1 and over the considered frame for c_2), the strategy c_1 provided smaller bitrate overhead and better MS-SSIM than c_2 on average. Compared to the size of latent tensor bitstreams, the size of quantized-OMPs bitstreams is only approximately 0.06% for the CLIC test set and 0.008% for the JVET-CTC sequences. Moreover, we evaluated the performance of the intra-frame

TABLE I: Experimental results.

Model	Data	BPP	MS-SSIM	Combined Size
NNVC	CLIC	0.03095	0.97347	1306MB
VVC8	CLIC	0.03487	0.97105	1298MB
RLVC-RGB	CLIC	0.06752	0.97056	2663MB
RLVC-YUV	CLIC	0.06752	0.97494	2663MB
E2E_T_OL	CLIC	0.03395	0.97167	1306MB
NNVC	JVET	0.03772	0.97001	2197MB
VVC8	JVET	0.03693	0.97461	1999MB
VVC	JVET	0.03788	0.98576	2051MB
RLVC-RGB	JVET	0.07332	0.96927	4120MB
RLVC-YUV	JVET	0.07332	0.98178	4120MB

TABLE II: Comparison between DSA and GSA blocks.

Model	BPP	MS-SSIM	Score	Parameters
DSA-2	0.13168	0.98857	0.97540	55M
DSA-3	0.13182	0.98883	0.97565	76M
GSA-2	0.13355	0.98837	0.97502	52M
GSA-3	0.13046	0.98843	0.97538	73M

TABLE III: Additional studies.

Intra codec only					
ID	**Overfit layers**	**Overfit frames**	**BPP**	**MS-SSIM**	**Score**
c_1	*Few*	*First*	0.11190	0.98550	0.96872
c_2	*All*	*All*	0.11196	0.98544	0.96864
c_3	*None*	*None*	0.11175	0.98539	0.96863
Inter codec only					
ID	**Overfit Combiner**	**Scaling motion**	**BPP**	**MS-SSIM**	**Score**
c_4	-	-	0.01201	0.98487	0.98367
c_5	-	✓	0.01913	0.99179	0.98988
c_6	✓	-	0.01216	0.98589	0.98468

codec without overfitting the OMPs (c_3 in Table III), which confirms the benefits of our proposed technique.

We also performed another ablation study involving the inter frame codec (see **Inter codec only** part in Table III), which was trained on a subset of the CLIC training set, and evaluated on 5 separate videos. We evaluated the contribution of overfitting the scaling parameters and of resolution adaptation. c_4 is our inter-frame codec without overfitting the Combiner's scaling parameters and without resolution-adaptation scaling of motion; c_5 is our inter-frame codec without overfitting the Combiner's scaling parameters but with resolution-adaptive scaling of motion; c_6 is our inter-frame codec with overfitting the Combiner's scaling parameters but without resolution-adaptive scaling of motion. $\lambda = 0.1$ was used to measure the Score. The ablation study results demonstrate that the proposed techniques clearly improve the coding performance.

IV. Conclusions

In this paper, we proposed a set of techniques for enabling a learned video codec to be highly adaptive to the input content, thus overcoming potential limitations caused by domain shift. Via extensive experiments, we showed the benefits of our techniques, and compared our codec to state-of-the-art video codecs in different settings and for different datasets.

References

[1] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, "Causal contextual prediction for learned image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[3] International Organization for Standardization, 2021, iSO/IEC 23090-3:2021 - Information technology — Coded representation of immersive media — Part 3: Versatile video coding.

[4] G. Sullivan, J.-R. Ohm, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, 12 2012.

[5] C.-Y. Wu, N. Singhal, and P. Krähenbühl, "Video compression through image interpolation," in *ECCV*, 2018.

[6] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with hierarchical quality and recurrent enhancement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," 2020.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[9] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 510–519.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[11] C. Aytekin, X. Ni, F. Cricri, J. Lainema, E. Aksu, and M. Hannuksela, "Block-optimized variable bit rate neural image compression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[12] C. Aytekin, F. Cricri, A. Hallapuro, J. Lainema, E. Aksu, and M. Hannuksela, "A compression objective and a cycle loss for neural image compression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[13] G. Lu, C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, and Z. Gao, "Content adaptive and error propagation aware deep video compression," *ArXiv*, vol. abs/2003.11282, 2020.

[14] J. Campos, S. Meierhans, A. Djelouah, and C. Schroers, "Content adaptive optimization for neural image compression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[15] N. Zou, H. Zhang, F. Cricri, H. R. Tavakoli, J. Lainema, M. Hannuksela, E. Aksu, and E. Rahtu, "L2c – learning to learn to compress," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 2020, pp. 1–6.

[16] Y. H. Lam, A. Zare, C. Aytekin, F. Cricri, J. Lainema, E. Aksu, and M. Hannuksela, "Compressing weight-updates for image artifacts removal neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[17] Y.-H. Lam, A. Zare, F. Cricri, J. Lainema, and M. Hannuksela, "Efficient adaptation of neural network filter for video compression," *arXiv:2007.14267 [eess]*, 2020.

[18] T. van Rozendaal, I. A. M. Huijben, and T. S. Cohen, "Overfitting for Fun and Profit: Instance-Adaptive Data Compression," *arXiv:2101.08687 [cs]*, Jan. 2021, arXiv: 2101.08687. [Online]. Available: http://arxiv.org/abs/2101.08687

[19] H. Liu, T. Chen, P. Guo, Q. Shen, X. Cao, Y. Wang, and Z. Ma, "Non-local attention optimized deep image compression," *arXiv:1904.09757 [cs, eess]*, Apr 2019. [Online]. Available: http://arxiv.org/abs/1904.09757

[20] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *arXiv:1809.02736 [cs]*, Sep 2018, arXiv: 1809.02736. [Online]. Available: http://arxiv.org/abs/1809.02736

[21] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *arXiv:1504.06852 [cs]*, May 2015. [Online]. Available: http://arxiv.org/abs/1504.06852

[22] H. Liu, M. Lu, Z. Ma, F. Wang, Z. Xie, X. Cao, and Y. Wang, "Neural video coding using multiscale motion compensation and spatiotemporal context model," *arXiv:2007.04574 [cs, eess]*, Jul 2020. [Online]. Available: http://arxiv.org/abs/2007.04574

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385 [cs]*, Dec 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[24] H. Zhang, F. Cricri, H. R. Tavakoli, N. Zou, E. Aksu, and M. M. Hannuksela, "Lossless Image Compression Using a Multi-Scale Progressive Statistical Model," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, Nov. 2020.

[25] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 10 771–10 780. [Online]. Available: http://papers.nips.cc/paper/8275-joint-autoregressive-and-hierarchical-priors-for-learned-image-compression.pdf

[26] R. Yang, F. Mentzer, L. Van Gool, and R. Timofte, "Learning for video compression with recurrent auto-encoder and recurrent probability model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 388–401, 2021.