REDUCED COMPLEXITY MULTISCALE CNN FOR IN-LOOP VIDEO RESTORATION

Kiran Misra, Andrew Segall, Byeongdoo Choi

Amazon Prime Video

ABSTRACT

Convolutional neural networks (CNNs) have shown promising improvements in video coding efficiency when included in traditional block-based codecs as a loop filter. Unfortunately, these coding gains are often accompanied by significant increases in complexity, measured by the number of multiply-accumulate (MAC) operations, that make them intractable in practice. As a result, there is considerable interest in reducing complexity for these CNN-based approaches. In previous work, we have shown that multiscale CNNs provide a path to reduce the associated MAC count. In this paper, we extend our work to consider channel grouping, spatial support limitations and shallower network depths to further reduce the MAC count of these multi-scale architectures. We demonstrate that the method can achieve an average VMAF bitrate reduction of 6.1% and 2.6% for all intra and random-access coding respectively, when compared to the evolving AV2 standard. Complexity is reduced to 1.85k MACs per pixel, which is a 390× reduction over previously published results.

***Index Terms*—** Multiscale Convolutional Neural Network, Video Compression, Restoration

1. INTRODUCTION

Video compression is a key enabling technology in the world today. It facilitates streaming video over the Internet in applications such as video on demand, live sports, game livestreaming and video conferencing – all of which saw explosive growth during recent years. Video coding standards, such as VP9 [1], AV1 [2], AVC [3], HEVC [4], VVC [5] are developed to enable interoperability between decoding devices. Each generation of video coding standard has approximately doubled in compression efficiency over the previous. Despite this rapid pace of improvement, video data currently represents more than 80% of consumer traffic on the Internet [6] and fuels the need for even higher video coding efficiency. Compression gains across the generations have typically been achieved by improving traditional video compression tools, such as transforms, quantization, motion compensation, and entropy coding.

Recent work has shown that incorporating CNN-based methods into video codecs as in-loop filters can also improve coded picture quality and achieve bit rate reductions [7]-[13]. While this is a promising area of work for the next generation of video coding standards, the bitrate reductions are often accompanied by a large increase in computational complexity. Specifically, the number of multiply-accumulate (MAC) operations may be in the millions of operations per input sample, resulting in complexities that are intractable for most decoders. As a result, there is significant interest in reducing the number MAC operations while keeping the bitrate reductions of these approaches.

In this paper, we build on our previous work on in-loop filtering using Multiscale Convolutional Neural Networks (MSCNNs) [13] by exploiting channel grouping to further reduce computational complexity. While channel grouping has been used to reduce complexity in other applications such as image classification [14], to our knowledge, this is the first investigation of its use within in-loop video restoration. As a result of the grouping, channels are partitioned into sets and processed largely independently within the neural network. This reduction in channel cross-talk directly translates to a reduction in MAC count. In addition to channel grouping, we explore limiting the spatial support of the convolutional kernels as well as decreasing network depths to further reduce the overall MAC count. The result is a MSCNN architecture with a MAC count of 1.85k per pixel. This is a 390× reduction compared to our previous work [13], while providing VMAF [15] bitrate reduction of 6.1% and 2.6% for all intra and random-access configurations, respectively.

The rest of the paper is organized as follows: Section 2 describes previous work in the area of in-loop filtering using CNNs. Section 3 provides details of our proposed model architecture. Section 4 describes the training and evaluation protocols. Section 5 includes experimental results. Section 6 provides an ablation study. Concluding remarks are provided in Section 7.

2. BACKGROUND

Typical CNN-based in-loop video restoration approaches make use of residual neural networks. Several variations have been explored in literature. For example, a depth-adaptive variant using residual CNNs is described in [8]. Wide activation residual networks that modify the structure of the residual block and vary its count are considered in [9]. Incorporating attention networks as well as inputs other than reconstructed samples, such as quantization parameters, boundary strength and coded partition information, is explored in [10]. Coding luma and chroma within a single CNN is considered in [11]. While these works demonstrate promising bitrate reduction, they all have MAC counts numbering in the hundreds of thousands, or millions of operations per pixel.

Previously, we explored a multiscale CNN architecture that processes a subset of channels at a resolution lower than that of the input picture [13]. Results show this is a promising direction to reduce computational complexity. In this paper, we build upon this multiscale CNN approach. Specifically, we further partition channels in the network into groups within the 2-D convolutional layers. The grouping reduces cross talk between channels. Additionally, we limit the spatial extent of the convolutional kernels and reduce the network depth to achieve an overall MAC count of 1.85k per pixel. This design represents a 390× reduction in per pixel MAC count over [13].

Note that other attempts to reduce the per pixel MAC count in video coding applications have also appeared in the literature. An architecture with 19k MAC operations per pixel is described in [7]. An approach that uses channel expansion and reductions to reduce MAC count to an estimated 5k MAC is described in [8], albeit with lower bitrate savings than reported here.

3. PROPOSED MODEL ARCHITECTURE

Figure 1 illustrates the placement of MSCNN within the AOMedia Video Model (AVM) codec. As shown in the Figure, the AVM may select between MSCNN, and the deblocking and constrained directional enhancement filter (CDEF) operations. This selection is indicated within the video bitstream at the picture level. And, when MSCNN is enabled for the picture, an additional 1-bit flag is signaled indicating if block level control is enabled for the picture. When block level control is enabled, the block size is determined at the encoder and transmitted in the bit-stream. Control block sizes can be 16×16, 32×32, 64×64 or 128×128 (in luma samples).

The method operates on the decoded luma sample values that are input to deblocking as well as dequantized inverse transform residue. It outputs a correction that is added to the input luma sample values to obtain a refined luma video signal. The correction samples values are scaled using scaling factors (1.0, 0.75, 0.50) that are transmitted at the picture level. This scaling enables an adaptation mechanism for content beyond those used to train the models.

For a picture, the encoder selects one of four pre-defined models available for a quantization parameter (QP) range and slice type. There is a total of six QP ranges. The six QP ranges chosen for Intra slice are: [0…100], [101…124], [125…149], [150…174], [175…200], [200…255] and for inter slice are: [0…110], [111…135], [136…160], [161…185], [185…210], [211…255]. The QP used for determining the model parameter values is derived as:

QP = (base qindex) - 24 * (source bit depth - 8)

where "base qindex" represents the base QP used for coding the video sequence.

Figure 2 shows the detailed model architecture for the MSCNN used in this paper and a visual example of the input-output. The example output shows that besides improving overall quality, the MSCNN also reduces blocking artifacts. The expression "conv2d kX nC sS" represents a two-dimensional 2-D convolution layer with a spatial kernel size of X×X, an output channel count of C, and a stride of S. (When not provided, S is one.) Similarly, "conv2d k(X,Y) nC sS" denotes the convolution kernel has spatial kernel dimensions X×Y. We denote convolutions using a diamond shaped filter as conv2dD. All convolutions are zero padded (as needed) within a 2-D convolutional layer to produce the same output spatial dimensions as the input. The residual blocks in Figure 2 have spatial kernel support that are either diamond-shaped, horizontal, vertical or 1×1. This is smaller than the traditional spatial kernel support of 3×3 in our previous work and helps reduce the overall per-pixel MAC complexity. There are five residual block stages.

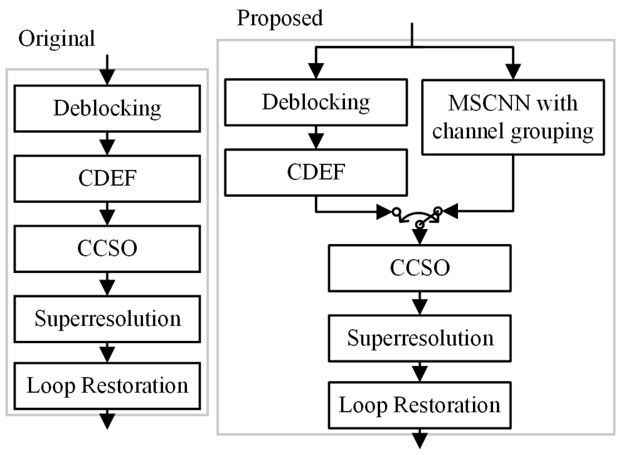

Figure 1: Placement of MSCNN with channel grouping within the AVM (tag: research-v3.0.0 [16])

Compared to [13], we rely on channel grouping to further reduce the overall number of MAC operations. Channel grouping is indicated by "gs G" in the "conv2d" notation, where G denotes the channel group size. The absence of "gs G" denotes that the convolution operates over all input channels and that the group size is the same as input channel count. A 2-D convolutional layer in Figure 2 performs $(G * C * KernelSpatialTapCount)/S$ MAC operations. As a result, when the input channel count, I, is greater than G, grouping of channels reduces the number of MACs contributed by a 2-D convolutional layer by a scale factor of I/G. Of the three processing paths shown in the Figure, the bottom path corresponds to I=28, G=8, which results in a 3× reduction of MACs.

In this paper, the input is initially converted to 31 channels. One of the channels is processed at full-resolution, while the remaining 30 are processed at one-half resolution. The full resolution processing path contains four residual blocks with diamond-shaped convolutions. The one-half resolution processing path (after the spatial downscaling) contains four residual blocks with alternating horizontal and vertical convolutions. These convolutions also use channel

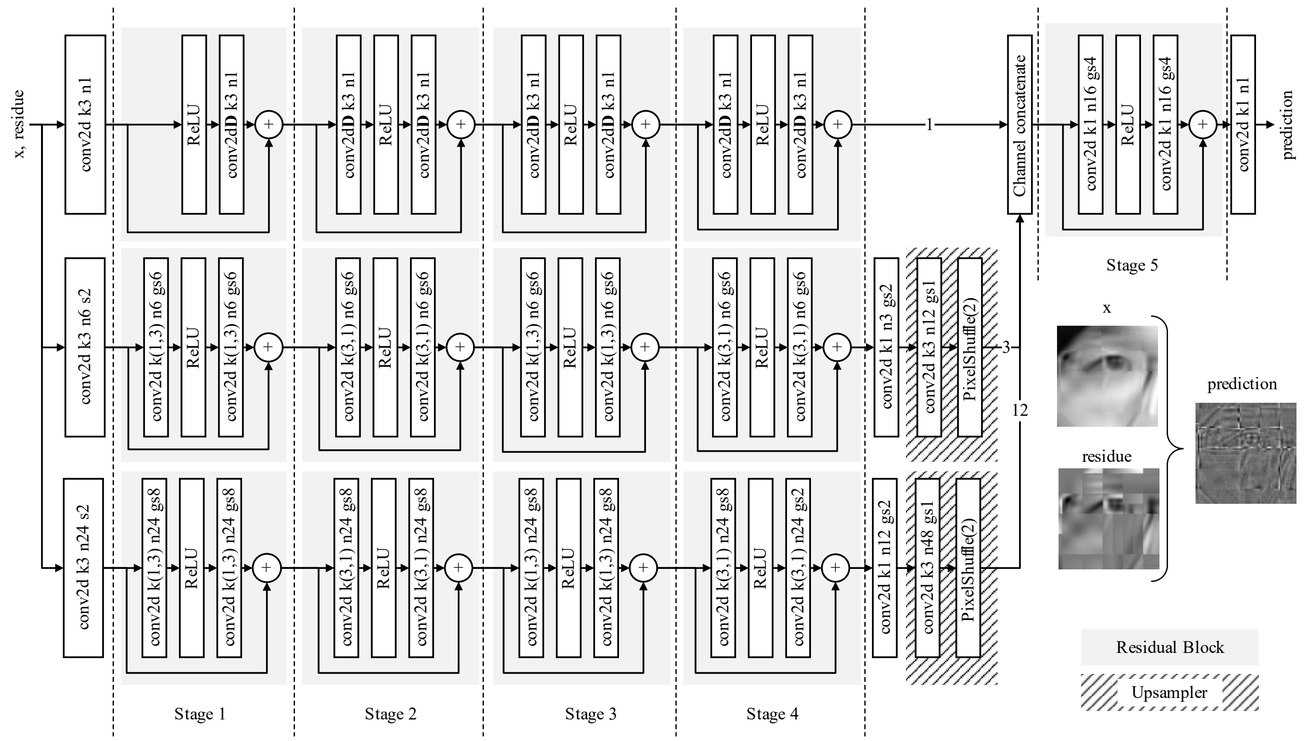

Figure 2: Multiscale CNN with channel grouping processing, 1 full resolution and 30 one-half resolution channels

groups, and the group sizes are 6, 8, 8, and 8, respectively. Operating at lower spatial resolutions, using input channel grouping, and reducing kernel spatial extent helps reduce the MAC per pixel associated with these layers. The overall spatial support region for MSCNN filtering is 23×23.

After processing by the residual blocks, the one-half resolution channels are fed to a 2-D convolution layer with input channel group size of two (i.e. 15 input channel groups) and spatial extent 1x1. The purpose of this 2-D convolution layer is to reduce the amount of data processed by subsequent layers, which is accomplished by producing one output channel for every two input channels. These 15 output channels are then upsampled independently and concatenated with the full-resolution channel. The concatenated result undergoes further processing by a residual block with 1x1 convolutions and group size of four. Finally, the result is processed with a 2-D convolution layer to generate the predicted correction for each luma sample. The overall per-pixel MAC count is 1,847.5. This is 390× lower than the per-pixel MAC count in [13].

4. TRAINING

The network uses four models for each combination of slice type and QP range. The intra slice type models are trained using the DIV2K dataset [17], while the inter slice type models are trained using the BVI-DVC dataset [18]. We use 256×256 patches with a batch size of eight. Intra model training is carried out for 3520 epochs with a learning rate of 10^{-3} for the first 3168 epochs, and 10^{-4} for the remaining epochs. For intra models, training data is generated by coding the DIV2K dataset using an all intra encoder configuration and the AVM research-3.0 software [16]. For inter models, training data is first generated by coding the BVI-DVC dataset using the trained intra models and a random-access configuration. Inter models are initially trained using this data for 320 epochs. We then re-generate training data using the partially trained inter model and continue training. During training, evaluation is carried out by using a subset of pictures from classes A2 and A3 of the AOMedia Common Test Conditions (CTC) [19]. Mean square error is used for the training loss.

5. EXPERIMENTAL RESULTS

The trained models are tested using AOMedia CTC for intra and random-access encoder configurations. The AVM software with tag research-v3.0.0 is used to conduct the experiments. In these conditions, test sequences are divided into multiple categories based on different use cases. We focus our efforts on camera-captured test sequences with resolutions class A1_4K (3840×2160,), class A2_2K (1920×1080), class A3_720p (1280×720), class A4_360p (640×360), class A5_270p (480×270). The source and model files used to perform the test are available at [20].

Table 1 reports the performance of MSCNN with channel grouping. As can be seen in the Table, the average VMAF Bjøntegaard Delta (BD) bitrate [21] improvement is 6.1% for the all intra configuration and 2.6% for the random-access configuration. We observe that the improvement is largely consistent between the VMAF and VMAF neg (nVMAF), which is a VMAF mode that is less sensitive to enhancement.

Table 1: Performance of MSCNN with channel grouping over AOMedia CTC

All Intra (PSNR BD Rate)				
Class	Y	YUV	VMAF	nVMAF
A1_4K	-2.54%	-2.26%	-7.19%	-6.38%
A2_2K	-2.84%	-2.57%	-5.88%	-5.29%
A3_720p	-3.38%	-3.08%	-6.00%	-5.40%
A4_360p	-2.67%	-2.46%	-5.62%	-4.97%
A5_270p	-2.21%	-2.01%	-5.85%	-5.20%
Average	-2.73%	-2.48%	-6.11%	-5.45%
Random-Access (PSNR BD Rate)				
Class	Y	YUV	VMAF	nVMAF
A1_4K	-2.59%	-2.29%	-4.37%	-4.27%
A2_2K	-2.68%	-2.42%	-3.17%	-3.20%
A3_720p	-3.03%	-2.71%	-2.26%	-2.67%
A4_360p	-2.70%	-2.44%	-2.06%	-2.31%
A5_270p	-2.24%	-2.00%	-0.90%	-1.56%
Average	-2.65%	-2.37%	-2.55%	-2.80%

As can be seen in the Table, the bit-rate savings are also sensitive to the video sequence resolution. For example, we observe (VMAF) coding improvement of 7.2% and 4.4% for UHD (4K) sequences in the all intra and random-access configurations, respectively. This is impactful in practice, as higher resolution sequences typically require larger bit-rates and correspondingly larger storage and transmission costs.

6. ABALATION STUDY

While the proposed architecture shows meaningful improvement in coding efficiency, we also provide results of our ablation study as an additional contribution of the paper. We examine how changes introduced in the model design impacts coding efficiency and per-pixel MAC complexity.

Table 2 reports on the tradeoff between complexity and coding performance for the different aspects under study. In the Table, S_0 represents channels processed at full-resolution and S_1 represents channels processed at one-half resolution (output by first 2-D convolutional layer). The average YUV BD bitrate corresponds to that of class A3, A4, A5 all-intra configuration. We use YUV BD bitrate in the ablation study to facilitate comparison with cases where the BD bitrate improvement in luma is accompanied by a loss in chroma (YUV results are also provided in Section 5).

As can be seen in the Table, it is possible to achieve different tradeoffs between coding performance and complexity. The highest performing configuration appears in the first row and corresponds to the performance of [13]. As can be seen, this method achieves a YUV bitrate savings of 6.6% at the complexity expense of 720k MAC operations per pixel. One straightforward way to reduce this complexity is to reduce the number of channels processed at each resolution and the number of residual blocks. When the number of channels is reduced (to (1,30)) along with the number of residual block stages (to 5), there is about 38.7× drop in per-pixel MAC count, while reducing bitrate savings to 3.7%. This is shown in the second row of the Table.

Limiting the spatial support of the convolutional kernels further reduces the per-pixel MAC count. This is shown in the third row of the Table, where complexity is reduced to 7.8k MAC operations (2.4×) and a bitrate savings of 3.3%. The final rows in Table 2 investigate the impact of channel grouping. Specifically, they show the impact of reducing the channel group sizes. In the fourth row, the 30 one-half resolutions are grouped in sizes of 6,8,8,8, representing an average group size reduction of 4×. The provided bitrate savings of 2.5% is interesting while reducing the per-pixel MAC count by 4.2×. In the final experiment, we employ 15 groups of 2 channels that represents a group size reduction of another 4× when compared to the row above. This drops the per-pixel MAC count even further, but the bitrate savings are now below 2%. As can be seen in the Table, reducing channel group size is effective in further reducing the MAC count.

7. CONCLUSION

In this paper, we report on the use of channel grouping to reduce the complexity of a multi-scale CNN architecture operating within a video codec. We demonstrate that with channel grouping, spatial support limitations and shallower depths, these multi-scale methods can achieve an average VMAF bitrate reduction of 6.1% and 2.6% for all intra and random-access coding configurations, respectively, when compared with the evolving AV2 standard. Complexity is reduced to 1.85k MAC per pixel. This represents a 390× reduction in per pixel MAC count over previous work. We assert that the proposed architecture is more suitable for use within video decoding devices that are complexity constrained.

Table 2: Complexity-coding performance tradeoff of various MSCNN designs

(S_0, S_1)	Residual block stages	Group size for S_0 | S_1	Limit kernel spatial support	Parameter count per model	MACs/pixel	Spatial extent	Average YUV BD rate
(32,96) [13]	9	32 | 96	No	2,073,601	720,752.0	57×57	-6.60%
(1, 30)	5	1 | 30	No	68,862	18,648.0	39×39	-3.66%
(1, 30)	5	1 | 30	Yes	25,630	7,816.0	23×23	-3.27%
(1, 30)	5	1 | 6,8,8,8	Yes	7,105	1,847.5	23×23	-2.52%
(1, 30)	5	1 | 2,…,2	Yes	3,151	840.0	23×23	-1.53%

8. REFERENCES

[1] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, R.S. Bultje, "The latest open-source video codec VP9 - an overview and preliminary results", *Picture Coding Symposium (PCS)*, December 2013.

[2] Y. Chen, D. Mukherjee, et. al, "An Overview of Core Coding Tools in the AV1 Video Codec," *Picture Coding Symposium (PCS)*, June 2018.

[3] D. Marpe, T. Wiegand and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 134-143, Aug. 2006.

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[5] B. Bross et al., "Overview of the Versatile Video Coding (VVC) Standard and its Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736-3764, Oct. 2021.

[6] CISCO, "Visual Networking Index Complete Forecast Highlights," 2021.

[7] U. Joshi, Y. Chen, D. Mukherjee, O. Guleryuz, S. Li and I. S. Chong, "Switchable CNN-Based Same-Resolution and Super-Resolution In-Loop Restoration for Next Generation Video Codecs," IEEE *International Conference on Image Processing (ICIP)*, pp. 1946-1950, 2022.

[8] D. Ding et al., "A CNN-based In-loop Filtering Approach for AV1 Video Codec," in *Picture Coding Symposium (PCS)*, pp. 1-5, 2019.

[9] G. Chen et al., "AV1 in-loop Filtering using a Wide-Activation Structured Residual Network," *IEEE International Conference on Image Processing (ICIP)*, pp. 1725-1729, 2019.

[10] Y. Li et al., "Deep In-loop filter with fixed point implementation," document JVET-AA0111, *27th JVET Meeting*, July 2023.

[11] L. Wang et al., "Neural network based in-loop filter with a single model", document JVET-AA0088, *27th JVET Meeting*, July 2023.

[12] J. Wang, G. Ding, D. Ding, D. Mukherjee, U. Joshi and Y. Chen, "Quadtree-based Guided CNN for AV1 In-loop Filtering," *IEEE International Conference on Image Processing (ICIP)*, pp. 3331-3335, 2022.

[13] K. Misra, A. Segall, B. D. Choi, "Multiscale convolutional neural networks for in-loop video restoration," to appear in *IEEE Data Compression Conference*, 2023.

[14] A. Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25, NIPS*, 2012

[15] VMAF Metric Library, libvmaf v2.1.1 Source: https://github.com/Netflix/vmaf/releases/tag/v2.1.1

[16] AOM Video Model, Online: https://gitlab.com/AOMediaCodec/avm

[17] R. Timofte et al., "NTIRE 2017 challenge on single image super resolution: methods and results," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017.

[18] D. Ma et al., "BVI-DVC: a training database for deep video compression," *IEEE Transactions on Multimedia*, vol. 24, pp. 3847-3858, 2022.

[19] X. Zhao et al., "AV2 Common Test Conditions and Performance Measurement," Testing Subgroup of AOMedia Codec Work Group.

[20] https://gitlab.com/kiranmisra/avm_mscnn/-/tree/research-mscnn Commit: 0c21545baebefdd0049cdff492bdc55e3f7ffe9c

[21] G. Bjøntegaard, "Calculation of average PSNR difference between RD curves," in ITU-T proposals, VCEG-M33, 2001.