Multiscale convolutional neural networks for in-loop video restoration

Kiran Misra, Andrew Segall, Byeongdoo Choi

Amazon Prime Video
2021, 7th Ave
Seattle, WA 98121, USA
{misrakir, asegall, bdchoi}@amazon.com

***Abstract**: Incorporating neural networks into a video codec as an in-loop filter has been shown to provide significant improvements in coding efficiency. Unfortunately, the computational complexity associated with the neural network, specifically the number of multiply-accumulate (MAC) operations, makes these approaches intractable in practice. In this paper, we consider using a multiscale approach to reduce complexity while maintaining coding efficiency. Experimental results demonstrate a 5.4× reduction in MAC operations while achieving an average bit rate savings of 6.4% and 6.3% for all intra and random access coding, respectively, when compared to the evolving AV2 standard. Ablation studies are also provided and show that the approach achieves all but 0.2% of the coding efficiency of full resolution processing.*

1. Introduction

Recent video coding standards, such as AV1 [1] and VVC [2], employ in-loop filters to improve coding efficiency. These filters increase the quality of each decoded picture and, since the filters are in-loop, propagate the improvements to subsequent frames using the motion compensation process. While AV1 and VVC do use sophisticated approaches, recent work has shown that leveraging residual neural networks can provide further coding efficiency improvements[3][4][11][14]. Unfortunately, these improvements come at the cost of significant computational complexity – typically on the order of hundreds of thousands or even millions of multiply-accumulate (MAC) operations per pixel.

Here, we consider the feasibility of using multiscale processing to reduce the number of MAC operations. We propose an approach that provides a 5.4× reduction in MACs by splitting the network into full resolution and low resolution components. Key features of our approach include: (a) an ability to operate some of the residual neural network channels at one-half the input spatial resolution, (b) an approach for re-combining the full resolution and one-half resolution channels using residual blocks, and (c) the inclusion of 1×1 layers in the low-resolution to manage spatial support. For convenience, we refer to the proposed approach as a multiscale convolutional neural network (MSCNN) throughout the paper.

We integrate the MSCNN into AOM Video Model (AVM) [10] that corresponds to the evolving AV2 standard. Results show the approach provides a bitrate savings of 6.4% and 6.3%, respectively, for the intra and random access configurations. We then ablate the implementation to examine the benefits of the different parts of the design. We assert that this work is not limited to AVM and should be applicable to other video codecs as well.

The rest of the paper is organized as follows: Section 2 identifies previous work and motivates our approach. Section 3 provides details of the proposed model. Section 4 describes the training, testing and evaluation protocol. It also describes the test results and provides visual examples of subjective improvement. Section 5 provides results of the ablation study. Concluding remarks appear in Section 6.

2. Background

For video coding applications, residual neural networks are traditionally employed as loop-filters, and researchers have explored variations of the network design for different performance objectives. For example, the authors of [3] consider depth adaptive convolutional networks. Alternatively, wide activation residual networks that modify the structure of a residual block and vary its count, are considered in [4]. Attention networks are also explored in [13]. Beyond the structure of the residual block, the use of additional information in the bit-stream has been shown to be beneficial and has been explored. For example, in [13], the quantization parameter, block strength and partitioning input are considered. Another approach where the authors also explore a network that processes luma and chroma information jointly and use quantization parameter as input is described in [12]. Finally, overtraining is considered in [15], where the biases of the network are overtrained for each intra-period and transmitted in the bit-stream.

The placement and relationship between the neural network and other loop filter processes has also been an area of study. In [13], it was proposed to have the network operate in parallel to the deblocking and sample-adaptive-offset (SAO) operations in the context of a VVC decoder. The result of the neural network and these other operations is then blended and subsequently processed by an adaptive loop filter. The blend factor is signaled on a block basis. Similarly, the authors in [12] consider a network that jointly processes luma and chroma information. In addition to controlling the blend factor on a block basis, the weights for the luma and chroma residual output are also signaled in the bit-stream. While most of the prior work places the neural network in the decoding loop so that the result is available for the prediction of future frames, an out-of-loop approach is considered in [15]. This means that the network operates after all other loop filters, and the result is not stored for prediction.

Unfortunately, the coding efficiency improvement of prior work is difficult to compare, since different researchers report results using different video codecs and evaluation conditions. However, it is reasonable to conclude that residual network loop filters typically provide coding efficiency improvements of between 5-13%. This though, comes at the cost of complexity. Specifically, a single convolutional layer of a residual neural network with C_{in} input channels, C_{out} output channels and kernel size K×K, performs $C_{in} \times C_{out} \times K \times K$ multiply-and-accumulate (MAC) for every spatial location of the output tensor. With multiple convolutional layers, the complexity of prior work may approach millions of MAC operations for each pixel within a picture. This requires specialized hardware and increased power requirements when operating at video data rates. And, as a result, we assert that the high MAC count of most residual neural network-based restoration approach makes them commercially prohibitive to implement within video decoder systems.

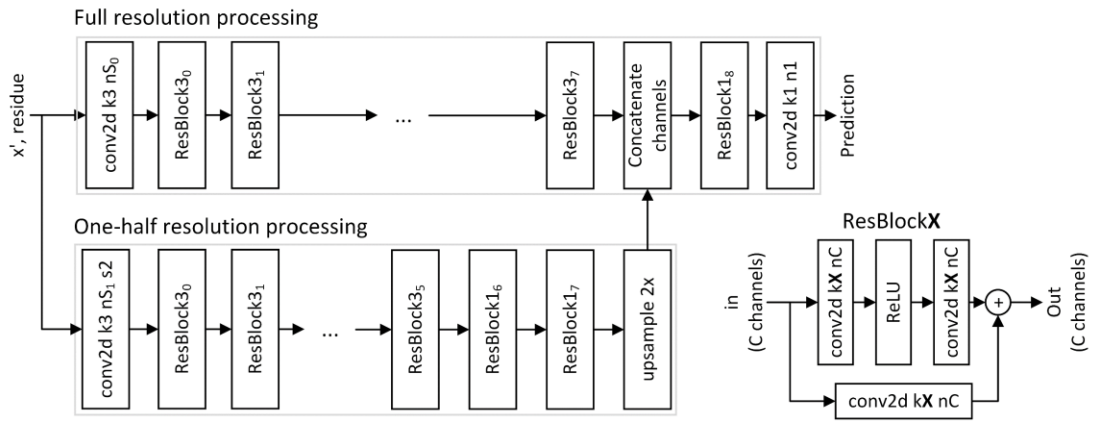

Figure 1: MultiScale Convolutional Neural Network (MSCNN) Architecture

While there are multiple dimensions that can be explored to reduce complexity, the focus of this paper is in the area of multiscale processing. We observe that while prior work may reduce the spatial dimensions of the input data when preparing it for input to the residual network, the spatial reduction is constant and fixed for all channels. Here, we investigate the question of whether some channels of a residual network can be processed at low-resolution while the remaining channels are processed at higher resolution. The goal is to reduce complexity while minimizing the loss in coding efficiency. We assert that the approach also provides a path for more understanding and intuition of network behavior.

3. Proposed Architecture

Figure 1 shows the proposed multiscale convolutional neural network (MSCNN) . It takes luma samples (x') and the transform residual as input. The output of the network corresponds to a per-sample correction value that is added to x' to recover the restored signal. The MSCNN processes information at two spatial scales – a full resolution path and a half-resolution path. The full resolution and one-half resolution paths carry S_0 (32) and S_1 (196) channels respectively. The spatial dimension of the tensor being processed at full resolution is the same as the spatial dimension of the input picture x', while the spatial dimension of the tensor being processed at one-half resolution is one-half the width and height of the input picture x'. The up-sampling at the end of one-half resolution processing path is performed using the nearest-neighbor algorithm. The full resolution processing path contains nine residual blocks, while the one-half resolution processing path contains eight residual blocks. The last residual block in the full-resolution processing path is used to combine the output of different scales, and as will be shown in section 5.2, is critical in maintaining the coding efficiency of MSCNN. Both the full and one-half resolution processing path are preceded by convolution blocks. Section 5.3, demonstrates that the convolution in the one-half resolution path helps judiciously repack information at lower resolution to preserve coding efficiency. For the generic residual block used in MSCNN, the skip path includes a convolution layer. In Figure 1, "conv2d kX nC" represents a two-dimensional convolution with spatial kernel size X×X, output channel count C, stride of 1. For remainder of the paper, stride S is only specified (using "sS") when it differs from 1.

We integrate MSCNN into AOMedia Video Model (AVM) [10]. For the paper, we use research-v3.0.0 tag of the software. Figure 2 shows the placement of the neural network

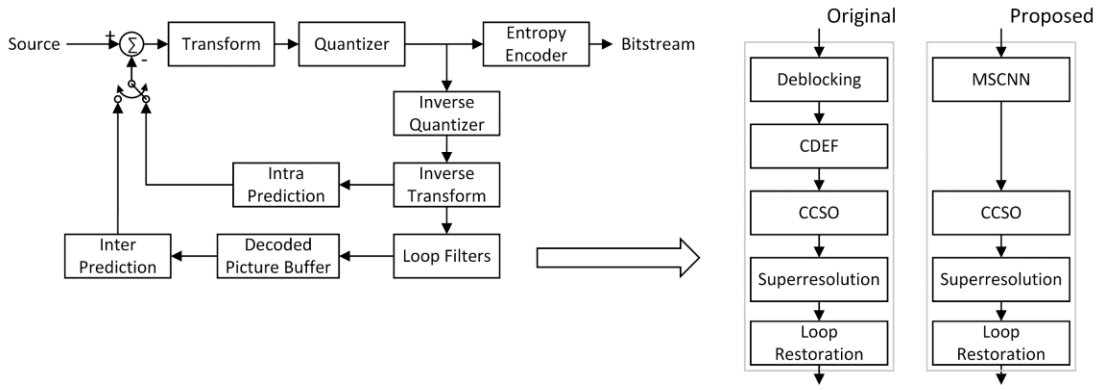

Figure 2: Placement of MSCNN in the AVM encoder (tag: research-v3.0.0)

within the video codec. As shown in Figure 2, the MSCNN replaces the deblocking and constrained directional enhancement filter (CDEF). The output of neural network is then provided to the cross-component sample offset (CCSO) block that uses luma samples to improve the fidelity of chroma components [9]. In the video system, we partition the quantization parameter (QP) space into 6 different ranges. A model is trained for each of the QP range. Moreover, different models are trained for intra and inter pictures. As a result, there are a total of 12 model (parameter sets). Model parameter values are switched at the picture level. The output of the model can be scaled by (1.0, 0.75, 0.5, 0) on a block-by-block basis. The block size can vary per picture and is selected from (16, 32, 64, 128).

4. Training, Evaluation and Test Protocol

To train the MSCNN, we use two different datasets. For intra pictures, we use the DIV2K dataset [5]; for inter pictures, we use the BVI-DVC dataset [6]. Training data is obtained by storing luma and transform residue sample values during encoding. The training protocol used for intra and inter models differ in the model initialization step. Intra model training uses random initialization, whereas inter uses pretrained models for initialization. Inter model training is iterative, and contains two-steps. In the first step, initialization is carried out using model parameter values of previously trained intra models. In the second step model parameter values obtained in first step is used for initialization. The loss function used for training is mean square error. More details are listed below:

Intra Model Training: For intra models, training data is generated by coding the DIV2K dataset with intra configuration of AOMedia Common Test Conditions (CTC) [8]. Next the training data is partitioned into 6 groups based on the following 8-bit QP ranges: [0…100], [101…124], [125…149], [150…174], [175…200], [200…255]. One model is trained using the data in each QP range. We use 1,760 epochs, a batch size of 1 and patch size of 256×256. Patches are selected to be aligned with a 128×128 grid boundary. For the first 90% of epochs the learning rate is 10^{-5}, and for the remaining epochs it is 10^{-6}. For each model, this translates to a training time of about 20 hours on a `g5.xlarge` instance.

Inter Model Training: For inter models, training data is generated by coding the BVI-DVC dataset with random access (RA) configuration of AOMedia CTC. Training data is partitioned into following 6 QP groups: [0…110], [111…135], [136…160], [161…185], [185…210], [211…255]. The bounds of each range are chosen to balance the number of pictures in each group. As described earlier in this section, inter model training is done in

Table 1: Performance of MSCNN over AOM CTC

Class	Intra (PSNR BD Rate)				Random Access (PSNR BD Rate)			
	Y	U	V	YUV	Y	U	V	YUV
A1 (4K)	-7.40%	4.13%	4.76%	-6.04%	-6.74%	7.52%	7.67%	-5.13%
A2 (2K)	-7.13%	3.06%	3.53%	-6.16%	-6.89%	4.71%	5.12%	-5.85%
A3 (720p)	-8.90%	3.12%	3.23%	-7.82%	-8.38%	3.81%	4.26%	-7.23%
A4 (360p)	-6.67%	3.79%	4.16%	-5.88%	-7.14%	2.91%	3.27%	-6.35%
A5 (270p)	-6.93%	2.61%	3.84%	-6.09%	-7.72%	0.62%	2.31%	-6.93%
Average	-7.41%	3.34%	3.90%	-6.40%	-7.38%	3.91%	4.53%	-6.30%

two steps. For both steps, training is carried out for 160 epochs. We use a learning rate of 10^{-6} for the first 90% of the epochs and 10^{-7} for the remaining. This corresponds to approximately 10 hours of training time on a `g5.xlarge` instance. For the first step, training data generation makes use of pretrained intra models. For the second step, training data generation makes use of pretrained intra models and inter models derived in first step.

Evaluation: For evaluation we select 26 pictures from video resolution classes A2 and A3 of AOMedia CTC. During evaluation we mask out 128×128 regions to simulate the block-level control used in the video coding system. After training, models with the best evaluation results are selected to be further tested in the AVM.

Testing: The trained models are evaluated using AOMedia CTC for Intra and RA configurations. AOMedia CTC defines a methodology on how to compare the performance changes of the coding tools in the context of the development of a next-generation video coding standard beyond AV1. In CTC, test sequences are divided into multiple categories based on different use cases. We focus our efforts on camera-captured test sequences with resolutions 4K (class A1: 3840×2160,), 1080p (class A2: 1920×1080), 720p (class A3: 1280×720), 360p (class A4: 640×360), 270p (class A5: 480×270).

Test Results: Table 1 lists the performance of MSCNN. The average YUV Bjøntegaard Delta (BD) bitrate [7] improvement is 6.4% for the intra configuration and 6.3% for the RA configuration. The complexity characteristics of the MSCNN are shown in Table 2. The network has approximately 2M parameters, 720k MAC operations per luma sample and a spatial extent of 57 pixels. We list spatial extent, since it impacts the number of line buffers that need to be stored for the processing algorithm. This complexity information will be further explored in the ablation studies reported in section 5. Figure 3 shows a cropped version of an inputs x' (from evaluation dataset) provided to MSCNN (QP 160). and sum of output "prediction" with input x'. As can be seen in Figure 3, MSCNN is effective in reducing the ringing artifacts, especially at the edge of the person's face. MSCNN also reduces blocking artifact, as can be observed in the hair area.

Table 2: Complexity-Coding performance of MSCNN

(S_0, S_1)	Parameter count	MACs/pixel	Spatial Extent	YUV BD Rate
(32, 96)	2,073,601	720,752	57x57	-6.60%

5. Ablation Study

In this section we investigate various design aspects of MSCNN. More specifically, we remove design features of MSCNN and report impact on performance and complexity.

(a) Input x' (b) Input x' + Prediction

Figure 3: Visual comparison of input and output of MSCNN

At a high-level, multiscale processing involves, processing S_0 channels at full-resolution and S_1 channels at one-half resolution. Each processing path contains a series of residual blocks. The one-half resolution processing path is preceded by a 2× downsampler and terminates in a 2× upsampler. The channels output by the upsampler are concatenated with full-resolution channels and undergo further processing to derive a prediction. With the help of ablation study performed in this section, we demonstrate that such a multiscale approach, with carefully chosen values for parameters S_0 and S_1 can reduce complexity while maintaining much of the coding gains. The coding gains used for complexity comparisons correspond to average Intra YUV BD bitrate of class A3, A4, A5 test sequences. The last column in Table 2 (and subsequent tables) correspond to this average.

Processing at multiscale is effective in reducing MACs/pixel count since the number of samples to be processed at lower resolution is smaller. For example, a convolutional layer used in one-half resolution processing path introduces a quarter of the MACs/pixel when compared to the same convolutional layer in full resolution processing path. While it may be tempting to perform all of the processing in lower resolution, doing so does not always reduce MACs/pixel complexity. The reason for this can be determined by examining the contribution of MACs/pixel from a convolution layer in one-half resolution processing path, with S_1 input channels, S_1 output channels and kernel size K×K i.e. $0.25 * (S_1 \times S_1 \times K \times K)$. While processing at one-half resolution reduces MACs/pixel linearly by 0.25, increasing S_1 increases the MACs/pixel and is proportional to S_1^2 which is a faster rate. As S_1 becomes larger, the square-increase overtakes the linear reduction afforded by spatial scale reduction.

We begin our study by examining a neural network that does not use multiscale aspect for restoration and does all its processing at full resolution (i.e. $S_1 = 0$). Section 5.1 describes one such network and lists its complexity-coding performance. Sections 5.2 examines the impact of removing residual block processing when merging the two resolution scales. Section 5.2 also examines the impact of 1×1 residual blocks replacing 3×3 residual blocks in the lower resolution processing path. Finally, in section 5.3, we examine the question if multiscale discards higher resolution information or simply judiciously repackages it.

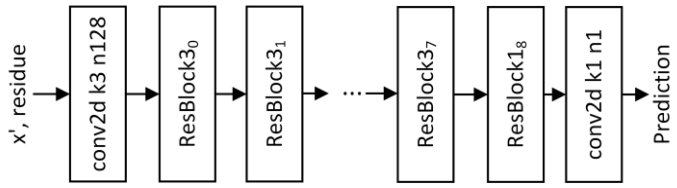

Figure 4: Architecture of residul network processing at full resolution only

5.1. Removing multiscale processing

Figure 4 shows the architecture diagram for the a neural network that processes all the information at full resolution. We refer to the neural network in Figure 4 as the Full Resolution Convolutional Neural Network. (FRCNN). FRCNN is made up of a 3×3 convolution followed by eight 3×3 residual blocks, a 1×1 residual block and a final convolutional layer that merges the various channels to output the prediction. The full resolution neural network has an internal channel count of 128 (same as in MSCNN). Table 3 below lists the complexity and coding efficiency results for the neural network in Figure 4. Compared to MSCNN the FRCNN neural network has 0.23% better compression efficiency but also has significantly larger MACs/pixel count, 5.4× of MSCNN. The increased MACs/pixel count does not justify the coding efficiency improvement. Note, the spatial extent of the neural network in Figure 4 is smaller than that of MSCNN. This is expected as a sample distance of 1 unit in one-half resolution tensor corresponds to 2 units in the original picture resolution. We address this aspect in section 5.2.

Table 3: Complexity-Coding performance when multiscale processing is removed

Parameter count	MACs/pixel	Spatial Extent	YUV BD Rate
3,594,113	3,590,528	35×35	-6.83%

5.2. Removing residual block processing when merging scales

In this section, we first consider an architecture where the 1x1 residual block processing is carried out before the two resolution scales are merged. Figure 5 shows the neural network architecture investigated. Here we merge by making use of a convolution layer. We examine the performance of the neural network in Figure 5 using different combinations of (S_0, S_1). Table 4 below contains the results of these experiments.

Table 4: Complexity-coding performance when using convolution to merge scales

(S_0, S_1)	Parameter count	MACs/pixel	Spatial Extent	YUV BD Rate
(64, 64)	1,800,065	1,123,712	71×71	-6.32%
(32, 96)	2,248,577	731,264	71×71	-5.69%
(16, 112)	2,809,217	745,280	71×71	-5.07%

The results show that as we push more channels to be processed at lower resolution the MACs/pixel (and the parameter) count drops until it reaches a certain point. Beyond this point the MACs/pixel (and the parameter) count starts to rise back up. The coding efficiency on the other hand shows a consistent and steady drop. While the results are promising for complexity reduction, the coding efficiency drop with respect to (w.r.t.) MSCNN and FRCNN is not trivial.

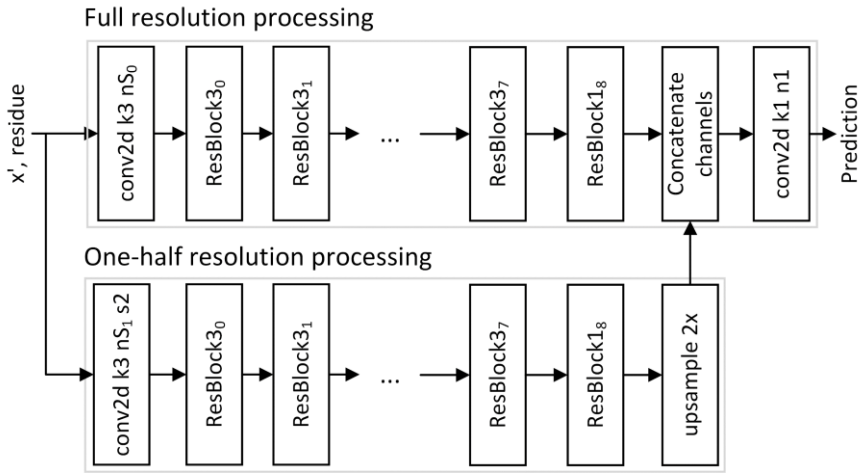

Figure 5: Merging scales using convolution

Next we consider a neural network where the different scales are merged using residual block with spatial extent 1×1 followed by a convolution. Figure 6 shows architecture diagram when using such an approach. The 1×1 residual block provides an opportunity for further non-linear adjustments to be made to the prediction based on the outputs of full resolution and one-half resolution processing path. This opportunity helps recover much of the coding efficiency losses observed in Table 4. While adding still more residual blocks can potentially improve the coding efficiency performance, it would also add complexity to the neural network. Table 5 lists the complexity and coding efficiency performance for the same set of (S_0, S_1) parameter values as in Table 4.

Table 5: Complexity-coding efficiency trade-off when using residual block to merge scales

(S_0, S_1)	Parameter count	MACs/pixel	Spatial Extent	YUV BD Rate
(64, 64)	1,824,641	1,157,504	71×71	-6.78%
(32, 96)	2,267,009	770,432	71×71	-6.57%
(16, 112)	2,819,969	784,256	71×71	-6.45%

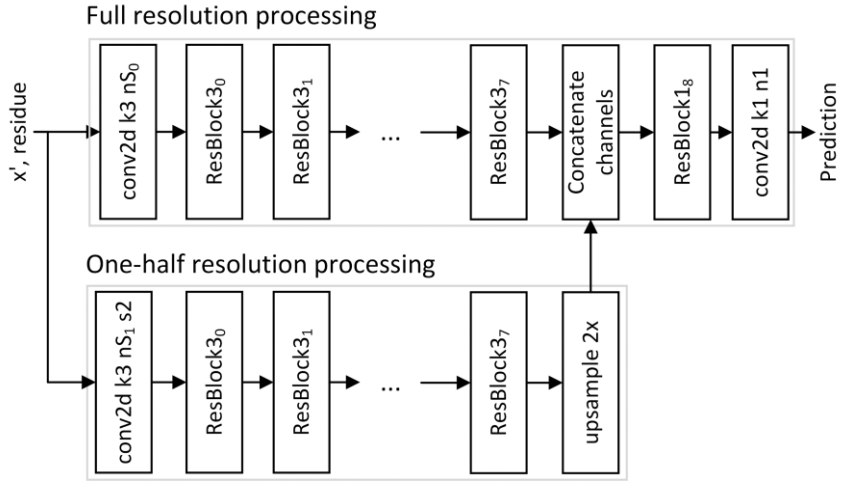

Figure 6: Merging scales using residual block and convolution

Comparing Table 4 and Table 5 we can see that the MACs/pixel reduction is smaller for the same values of (S_0, S_1) when the 1×1 residual block is moved to the full processing path. However, the coding loss compared to FRCNN of section 5.1 is much smaller as well. For (S_0, S_1) equal to (32, 96) the loss is 0.26% while reducing the MACs/pixel by about 5.4× which is an attractive tradeoff. Note, this choice of (S_0, S_1) also explains the coding efficiency difference between full resolution FRCNN of section 5.1 and MSCNN. As can be seen from Table 6, choosing (S_0, S_1) equal to (64,64) would have further reduced the coding efficiency gap w.r.t. FRCNN to 0.05% but at lower MACs/pixel reduction.

Table 6: Complexity-coding performance when converting ResBlock3_a to ResBlock3_b into ResBlock1 in one-half resolution processing path

(S_0, S_1)	(a, b)	Parameter count	MACs/Pixel	Spatial Extent	YUV BD Rate
(32, 96)	(6, 7)	1,824,641	659,840	63×63	-6.60%
(32, 96)	(5, 7)	1,603,457	604,544	55×55	-6.50%
(32, 96)	(4, 7)	1,382,273	549,248	47×47	-6.49%

While, the (S_0, S_1) equal to (32, 96) configuration of Table 6 provides an attractive MACs/pixel reduction it does require a larger spatial extent compared to FRCNN. To address this issue, we investigate reducing the spatial extent in one-half resolution processing path by converting the ResBlock3 building blocks (with 3×3 spatial extent convolutions) to ResBlock1 (with 1×1 spatial extent convolutions). Each such conversion reduces the overall spatial extent by 8×8. We convert the residual blocks starting from the back of the one-half resolution processing path (i.e. ResBlock3_7, ResBlock3_6, … in that order). Table 6 lists the complexity versus coding efficiency tradeoff when 2, 3, 4 such residual blocks are converted. Converting 2 of the residual blocks has almost no impact on coding efficiency (which is the design choice made for MSCNN). Reducing beyond 2 residual blocks reduces spatial extent but is accompanied by some coding efficiency loss. The coding loss indicates that the larger spatial extent of multiscale processing does play a role in mitigating some of the losses incurred when processing channels at lower resolution.

5.3. Performance when spatial decimation is performed prior to convolution in one-half resolution processing path

An important question that deserves some attention is whether the multiscale neural network is merely processing at lower resolution, features that can be subsampled and has no impact on the eventual restoration, or is packing information within channels of the one-half resolution processing path. To answer this question, we conduct a simple experiment where the first convolutional layer of MSCNN in one-half resolution processing path is preceded by a 2× decimation layer. The stride of first convolutional layer in the one-half resolution processing path is then set to 1. The average coding efficiency for such a configuration is shown in Table 7. It shows a loss in performance of about 0.14% w.r.t. MSCNN indicating that the first convolution in the one-half resolution processing path of the model shown in Figure 1 is indeed judiciously packing full-resolution information that it needs to preserve within the channels.

Table 7: Coding performance when one-half resolution processing is preceded by 2× decimation

(S_0, S_1)	Parameter count	MACs/pixel	Spatial Extent	YUV BD Rate
(32, 96)	1,824,641	658,544	57×57	-6.46%

6. Concluding Remarks

We present a multiscale convolutional neural network (MSCNN) architecture and study its application as a loop filter within a video codec. The architecture allows for the explicit processing of video data at multiple resolutions by a series of residual blocks. When compared to a similar network that only operates at the high resolution, we observe the multiscale approach reduces complexity by 5.4×. Results show that the network provides a bit-rate savings of 6.4% and 6.3%, respectively, for intra and random access configurations when integrated into the evolving AV2 video codec.

References

[1] Y. Chen et al., "An overview of coding tools in AV1: the first video codec from the alliance for open media", *APSIPA Trans. on Signal and Information Processing*, vol. 9, pp. e6, 2020.

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

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

[4] 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

[5] 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.

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

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

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

[9] X. Zhao et al., "Study On Coding Tools Beyond AV1," *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1-6, 2021.

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

[11] A. Segall, "Overview of technologies considered in JVET's neural network-based video coding exploration", document JVET-Y0239, 25th JVET Meeting, Jan. 2022.

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

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

[14] E. Alshina et. al., "Summary of Exploration Experiments on Neural Network-based video coding", document JVET-AB0023, 28th JVET Meeting, Oct. 2023.

[15] M. Santamaria et al., "Content-adaptive post-filter with SADL inference and signalling of NN post-filter characteristics and activation SEI messages", document JVET-AB0048, 28th JVET Meeting, Oct. 2023.