T. Partanen, A. Mercat, J. Vanne, M. M. Hannuksela, H. Zhang, A. Aminlou, and F. Cricri, "Adaptive luma range optimization in visual coding for machines", Third IEEE Workshop on Coding for Machines, Sep. 2025.

ADAPTIVE LUMA RANGE OPTIMIZATION IN VISUAL CODING FOR MACHINES

Tero Partanen[1], Alexandre Mercat[1], Jarno Vanne[1],
Miska M. Hannuksela[2], Honglei Zhang[2], Alireza Aminlou[2], and Francesco Cricri[2]

[1]Ultra Video Group, Tampere University, Finland
[2]Nokia Technologies, Finland
{tero.partanen, alexandre.mercat, jarno.vanne}@tuni.fi,
miska.hannuksela, honglei.1.zhang, alireza.aminlou, francesco.cricri}@nokia.com

ABSTRACT

The growing prevalence of machine-driven visual data consumption underscores the need to meet the unique requirements of Video Coding for Machines (VCM). In this paper, we propose to enhance the coding efficiency of Versatile Video Coding (VVC) for machine consumption by adaptively adjusting the dynamic range of the input luma channel prior to encoding. The visual input is characterized using the introduced input analyzer that predicts the optimal dynamic range and provides the corresponding 1) luma down-scaling factors applied before encoding and 2) luma up-scaling factors used after decoding to restore the dynamic range. Our input analyzer is implemented as a lightweight neural network. For the network training, we introduce a training framework incorporating a codec proxy module that enables end-to-end optimization by simulating a conventional non-differentiable video codec. The proposed method has been evaluated as part of the conventional VVC pipeline, where VVC test model (VTM) is used for encoding and decoding. Our experimental results show that integrating the proposed solution into the pipeline improves coding efficiency by up to 28.0% on image datasets and up to 45.4% on video dataset for object detection tasks.

Index Terms—Video Coding for Machines (VCM), Machine Vision, Neural Networks (NN), Versatile Video Coding (VVC)

1 INTRODUCTION

Machine vision-based visual data analysis has become an integral part of numerous applications, including smart mobility and transportation, security and surveillance, healthcare diagnostics, and industrial automation. These applications critically rely on the quality and integrity of the visual data they use, necessitating efficient compression schemes that preserve machine task accuracy within the limits of available bandwidth and storage space. The *Versatile Video Coding* (*VVC*) standard [1] represents the cutting edge of visual data compression, approximately doubling the coding efficiency of its predecessor, *High Efficiency Video Coding* (*HEVC*) [2]. However, these conventional video coding standards are primarily optimized for human visual perception, so they tend to yield bitrate overhead in machine vision applications [3]. To that end, there is an urgent need to adapt modern visual coding schemes for machine consumption.

This work was supported by the Research Council of Finland (decision no. 349216).

Recent standardization activities by the *Moving Picture Experts Group* (*MPEG*) [4] and *Joint Video Experts Team* (*JVET*) [5] seek to address the emerging field of machine-oriented image and video coding. This domain is typically referred to as *video coding for machines* (*VCM*) [6]. The fundamental objective of VCM is to develop image and video coding tools that are specifically optimized for machine-based analysis or hybrid machine-human consumption.

Existing solutions implemented within conventional video coding schemes, such as VVC, typically involve pre-processing, encoding optimization, and post-processing techniques [5]. The techniques for pre-processing and encoding optimization are predominantly built on *region of interest* (*ROI*)-based methods [7]–[10]. Alternatively, pre-processing may include spatial downsampling [11], [12] or dynamic range scaling of pixel intensities [13], wherein the dynamic luma range is downscaled by multiplying all luma values by a predetermined factor. In contrast, post-processing techniques mainly include various compression artifact suppression filters [14], [15], which are tailored to improve signal fidelity for machine analysis tasks.

In this work, we propose an adaptive luma range optimization method for VCM. It incorporates an input analyzer module that analyzes the given input prior to encoding and predicts optimal luma down-scaling and up-scaling factors. Our previous approach [13] applied a predetermined and constant luma scaling factor across all content, whereas the proposed method dynamically adapts to the visual characteristics of the given input data. This work also introduces a novel training framework for the input analyzer. The framework incorporates a codec proxy to enable end-to-end optimization by simulating a conventional, non-differentiable video coding pipeline.

The remainder of this paper is organized as follows. Section 2 provides an overview of related pre-processing and post-processing techniques for VCM. Section 3 describes the proposed coding pipeline, particularly focusing on the designed input analyzer module and the novel training framework introduced for it. The experimental setup is presented in Section 4 and experimental results in Section 5. The conclusions are given in Section 6.

2 RELATED WORKS

Pre-processing and encoding optimization techniques developed for VCM primarily aim at bitrate reduction. They are typically implemented through ROI methods that initially analyze input data to identify regions considered salient for downstream machine tasks [5], [7]. The existing ROI-based pre-processing methods achieve coding gains by either blurring [9] or completely

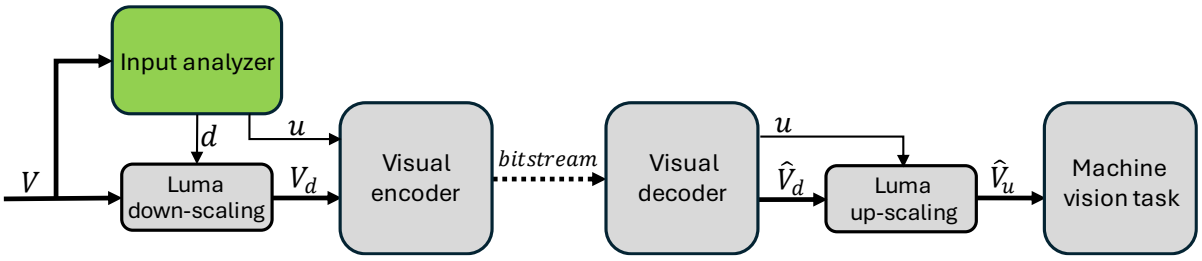

Fig. 1. Proposed end-to-end coding pipeline, where the core component is highlighted in green.

suppressing [10] (non-ROI) background regions. The encoding optimization techniques, on the other hand, use saliency information to guide the encoder to allocate more bits to ROI regions than to non-ROI regions [8], [16]–[18]. A principal limitation of ROI-based methods lies in their optimization for object-based machine tasks, where only discrete objects are considered as ROIs. ROI-optimized compression can lead to substantial accuracy degradation in machine tasks that require access to non-ROI regions. Pre-processing approaches independent of ROI techniques include, e.g., spatial downscaling of input frames. The applied input downscaling factor may be predetermined [11] or adaptively estimated [12]. Additionally, such approaches may incorporate a post-processing step in which the decoded visual data is upscaled to its original resolution prior to the machine task [12].

While encoder-side techniques are designed to reduce bitrate, the decoder-side post-processing techniques seek to enhance the quality of the reconstructed data for machine tasks. These post-processing techniques predominantly concentrate on deep learning-based quality enhancement filters [14], [15].

A key advantage of decoder-side techniques lies in their ability to leverage learning-based methods through joint optimization with neural network-based machine analysis tasks. Contrary, encoder-side techniques face challenges in joint, or so-called end-to-end, optimization due to the non-differentiable nature of conventional codecs, such as HEVC and VVC. To overcome this challenge, differentiable proxy networks have been proposed [19]. The function of such proxy networks is to approximate compression artifacts of the original codec and to provide back-propagation path from the decoder-side to the encoder-side, thereby enabling an approximated end-to-end optimization including both bitrate and task accuracy. In this approach, the effectiveness of joint optimization is highly dependent on the approximation accuracy of the proxy network, including the approximation of both compression-induced distortion and bitrate estimation. Another significant challenge is the limited availability of proxy networks for most conventional codecs. Both ROI-based methods and post-filtering approaches are typically implemented using deep learning models, such as *convolutional neural networks* (*CNN*), which can introduce significant computational overhead.

Our recent luma range scaling technique [13] has shown potential to significantly improve coding efficiency for VCM, while maintaining low computational overhead. In this technique, the dynamic range of a luma component is scaled down, prior to encoding, using a constant scaling factor, and optionally fully scaled back in an up-scaling step after decoding. While promising, this static approach lacks adaptability to diverse video content and encoding conditions. This current work overcomes these limitations by proposing an adaptive luma range optimization method that selects scaling strategies based on input analysis prior to encoding.

3 PROPOSED ADAPTIVE LUMA RANGE OPTIMIZATION

In general, scaling down the dynamic range of the luma component leads to a reduced bitrate; however, it tends to increase compression-induced distortions in the reconstructed signal, which potentially degrade machine task accuracy. One of the primary sources of these distortions is rounding errors caused by the limited precision of the integer-based arithmetic employed in typical codec implementations. In contrast, luma up-scaling impacts only the reconstructed signal, potentially amplifying existing distortions, and thereby affecting task accuracy without altering the bitrate. Results of our prior work [13] indicated that the optimal down-scaling factor may depend on the characteristics and content of the input. This also applied to the up-scaling factor, which potentially has an optimal value between no up-scaling and full up-scaling. To address the challenge of determining optimal scaling factors, we introduce our proposed adaptive method in the following sections.

3.1 Proposed Coding Pipeline

Fig 1 illustrates how the proposed adaptive luma range optimization method is incorporated into the complete coding pipeline. The pipeline accepts an input image or video (V) in the $Y'C_BC_R$ color space. The input V is first analyzed by the *input analyzer* module that seeks to yield an optimal trade-off between bitrate and machine task accuracy by predicting the optimal combination of luma down-scaling factor (d) and luma up-scaling factor (u) for the input, where $d \in (0, 1]$ and $u \in [1, 1/d]$. Subsequently, the *luma down-scaling* module multiplies the pixel values of the luma channel in V by d and feeds the luma down-scaled input (V_d) to the encoder. The encoded bitstream is then generated by the *visual encoder* from V_d along with u. The value of u may be encoded into the bitstream as metadata, e.g., by using *supplemental enhancement information* (*SEI*) message [13].

On the receiving end, the bitstream is decoded by the *visual decoder* that yields the reconstructed visual data ($\hat{V}_d$) and u. The *luma up-scaling* module multiplies the luma channel of $\hat{V}_d$ by u, resulting in the final reconstructed image/video ($\hat{V}_u$) to be consumed by machine vision tasks.

3.2 Proposed Training Framework

Fig. 2 illustrates the proposed training framework for the *input analyzer* module. End-to-end optimization is enabled by replacing the non-differentiable original codec with the *codec proxy*, which encompasses two main components:

1) **Residual injector**, which utilizes a dataset comprising pre-encoded and reconstructed versions of the training dataset, generated using numerous combinations of down-scaling

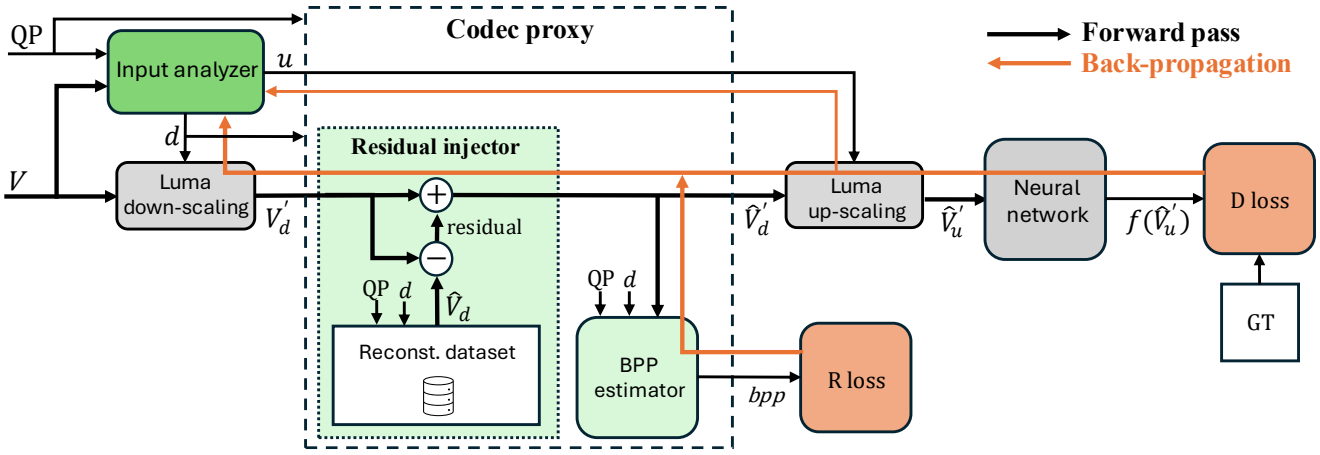

Fig. 2. Overview of the training framework for the input analyzer. The main contributions are highlighted in green.

factors d and *quantization parameter* (*QP*) values. Each training image in such dataset is associated with a specific d and QP value. During the forward pass, the reconstructed version of the input image, $\hat{V}_d$, is retrieved from the dataset based on the predicted d and QP. The residual, representing coding artifacts, is computed as the difference between $\hat{V}_d$ and the downscaled input image V_d'. The residual is then added to V_d' on the main forward path, resulting in $\hat{V}_d'$. The $\hat{V}_d'$ accurately replicates the reconstructed output of the original codec, including the distortions. The only minor difference is the use of floating-point arithmetic during training, as indicated by the prime symbol ($'$). Conclusively, the main data path through *residual injector* remains compatible with back-propagation, enabling end-to-end training.

2) **Bits-per-pixel (BPP) estimator**, which is a neural network used to provide an estimate of the rate loss of the simulated codec. By minimizing this rate loss, gradients can be computed and backpropagated to train the *input analyzer*. This estimator is pre-trained using the reconstructed dataset described above and is fully optimized (i.e., overfitted) specifically to that dataset to ensure accurate rate estimations. As a result of using the BPP estimator during training, the input analyzer is trained to minimize the estimated rate.

The two primary inputs of the training framework consist of the training image V and the corresponding QP value. The QP value is the same parameter used as input by the conventional encoders to set the baseline *rate-distortion* (*RD*) trade-off. In the training loop, V and the QP value are initially passed into the *input analyzer*, which outputs the down-scaling factor d and up-scaling factor u. At the output of the *input analyzer*, the value of d is discretized, e.g., by rounding, to align with a predefined set of discrete values corresponding to the down-scaling factors used in generating the reconstructed dataset within the residual injector. The luma channel of V is then down-scaled based on d, and the residual is added into the resulting V_d' by the *residual injector* as previously described.

Subsequently, the pre-trained *BPP estimator* takes $\hat{V}_d'$, d, and the QP value as inputs for analysis to estimate the *bpp* value, which constitutes an estimate of the rate loss R. Concurrently, the luma channel of $\hat{V}_d'$ is scaled up using the predicted up-scaling factor u, resulting in the representation of the up-scaled reconstructed image $\hat{V}_u'$. Next, this image is processed by the task *neural network* (*NN*) model f, which produces the output $f(\hat{V}_u')$. Then, the task loss D is computed from $f(\hat{V}_u')$ and a ground truth (GT) label using f-specific loss function ℓ_f [20]. Finally, the *input analyzer* is optimized using the total loss function

$$\mathcal{L} = R + \lambda_{QP} D \tag{1}$$

where λ_{QP} is QP-specific multiplier employed to control the RD trade-off in the training process. The λ_{QP} is a critical hyperparameter for achieving optimal learning and loss convergence during training. It may depend on the employed model f and requires empirical analysis for proper determination.

During the back-propagation step, gradients are calculated for the model f, the *BPP estimator*, the scaling modules, and ultimately for the *input analyzer*. However, only the parameters of the *input analyzer* are optimized, while the parameters of the model f and the *BPP estimator* remain fixed.

It is noteworthy that virtually any differentiable pre-trained NN model may be employed as the model f. Furthermore, multiple models can be incorporated in parallel, each with their own weighted loss functions and GT labels. The choice of models and the corresponding loss functions ℓ_f is guided by the intended use case. For instance, task-specific training can be derived by employing networks trained for that specific task such as *object detection* (*OD*) or semantic segmentation [21]. Alternatively, to support more general-purpose applications, multiple models trained for different tasks can be employed during training. Additionally, backbone networks [22] may be used in conjuction with feature loss, or a specific semantic-loss model [23] can be applied.

4 EXPERIMENTAL SETUP

Our experimental procedure consisted of the following steps. First, the *BPP estimator* was fully optimized—intentionally overfitted—on the reconstructed dataset. Next, the *input analyzer* was trained utilizing the described training framework. Finally, the evaluations were conducted. All experiments were carried out utilizing the PyTorch framework (version 2.4) along with Torchvision library (version 0.19.0) [24]. The *VVC test model* (*VTM*) version 20.0 [25] was employed for encoding and decoding.

4.1 Training Setup

Our training dataset was composed of 15 000 images that were randomly selected from the OpenImages V6 dataset [26], ensuring that none of them overlap with those in the evaluation datasets. To

construct the encoded and reconstructed dataset for the *residual injector*, the selected images were encoded using different combinations of down-scaling factors $d \in \{0.1, 0.2, \ldots, 1.0\}$ and QPs $\in \{22, 27, 32, 37, 42, 47\}$. This process resulted in a dataset totalling 900k images.

For the *BPP estimator*, pre-trained ShuffleNet V2 2X CNN architecture [27] from PyTorch's Torchvision library was adopted as the backbone. A single neuron regression-head was appended to the backbone to produce the BPP value output. The input images were resized to 640×640 at the backbone input. The QP value and d were each broadcasted to size of 640×640 and appended as fourth and fifth channels to the input image. In the *BPP estimator* training, the AdamW optimizer was used with an initial learning rate of 0.0001, decayed by 0.96 after each epoch. The model was trained for 200 epochs using the reconstructed dataset described above and subsequently evaluated on the same training data, resulting in an average BPP prediction error of 1.68%.

To support practical and real-time use cases, a light-weight pre-trained ShuffleNet V2 0.5X CNN architecture from Torchvision library was selected as the backbone for the *input analyzer*. The implemented regression head consisted of one hidden layer with 500 neurons, followed by a ReLU activation function and a two-neuron output layer. The input images were resized to 512×512 at the backbone input, and the QP-input was broadcasted to 512×512 and appended as a fourth channel to the input image. Since the original image resolution may influence the RD trade-off in compression, the original image width and height were also provided as additional input to the regression head to be factored into the prediction. During training, the output was clamped to the range [0.1, 1.0] and rounded to one decimal precision. During inference, the rounding was omitted.

In order to generalize the predictions of the *input analyzer* across various machine vision tasks, three distinct pre-trained NN models were employed to generate the task loss D: RetinaNet with ResNet-50 backbone [24], Faster R-CNN with MobileNetV3L backbone [24], and ResNetV2-101 backbone [28]. The first two of them are object detection models that provide task loss, while the last one is a backbone that provides a more general feature loss.

The RGB input was converted to YUV color space before feeding it to the *luma down-scaling* module, and back again to RGB at the output of the *luma up-scaling* module. These conversions were done according to the ITU-R BT.601 standard.

Training of the *input analyzer* was conducted for 8 epochs with a batch size of 4. The AdamW optimizer was initialized with a learning rate of 0.001, which decayed by 0.8 after each epoch. For the reported results, training was conducted using only QP values of 32, 37, and 42.

4.2 Evaluation Setup

The evaluations were conducted by simulating the proposed coding pipeline depicted in Fig. 1, in accordance with the JVET *common test conditions* (*CTC*) for VCM [29], and utilizing the *Bjontegaard Delta Bitrate* (*BD-rate*) [30] as the primary evaluation metric. The proposed method was compared with our prior work [13], which utilized fixed luma down-scaling factors d, combined with either *no up-scaling* (*NU*) or *full up-scaling* (*FU*). The anchor results for both evaluated methods were generated using the VTM for encoding and decoding without any external optimization techniques.

The performance assessesment was carried out across diverse content by focusing on three image datasets specified in the CTC: OpenImages V6 for OD, OpenImages V6 for *instance segmentation* (*IS*), each comprising 5000 images, and TVD-I [31], which contains 166 images and was evaluated for both OD and IS tasks. The applied coding configuration was *all intra* (*AI*). The machine vision architectures for above tasks are: Faster R-CNN with ResNext-101 backbone for OD and Mask R-CNN with ResNext-101 backbone for IS [32].

Table 1. BD-rate (%) results for image datasets. Our method and [13] using fixed d combined with no (NU) or full up-scaling (FU)

Method		OpenImages OD		OpenImages IS		TVD-I OD		TVD-I IS	
	d	NU	FU	NU	FU	NU	FU	NU	FU
	0.2	125.8	22.2	111.8	29.8	177.8	30.2	140.6	41.7
	0.3	57.9	-5.6	45.8	5.1	111.3	8.6	59.9	13.3
	0.4	32.8	-6.4	17.7	-2.6	60.6	2.8	65	10.8
[13]	0.5	13.1	**-7.4**	9.2	-4.4	36.9	1.2	56.6	8.7
	0.6	8.4	-3.8	3.3	-1.9	17.3	0.9	51.3	5.8
	0.7	0.8	-4.6	-1.0	-2.1	9.1	-4.1	38.3	**0.9**
	0.8	0.7	-2.1	-4.9	**-5.3**	11.6	2.2	15.5	4.1
	0.9	-5.3	-6.4	-2.0	1.6	1.4	**-4.3**	8.32	6.1
Our		**-28.0**		**-23.3**		**-21.1**		**-16.6**	

The SFU-HW-objects-v1 [33] (SFU) video dataset was utilized as a complementary assessment. The applied coding configuration included AI, *low* delay (*LD*), and *random access* (*RA*). In this evaluation, the *input analyzer*'s down-scaling prediction was applied only to the first frame of each test sequence, and the predicted scaling factor was then fixed for the remainder of the sequence, while up-scaling was applied frame-by-frame basis. This choice was made because modifying luma values for each input frame prior to encoding could potentially deteriorate the performance of inter prediction.

5 EXPERIMENTAL RESULTS

Table 1 reports the BD-rate results of our current and prior [13] methods relative to the anchor. The values, expressed as percentages, were measured at equivalent task accuracy levels. A negative BD-rate value indicates a reduction in bitrate, i.e., coding gain.

The results show that applying our prior method without up-scaling (NU) generally increases bitrate for most values of d. The best BD-rate improvement of this setup was -5.3% with d=0.9 in OpenImages dataset for the OD task. Full up-scaling (FU) improved the average results in all reported test cases, but the observed highest gain remains still moderate, being -7.4% with d=0.5 on the same dataset and task. Notably, on the TVD-I dataset with the IS task, our prior method resulted in coding overhead in all test cases, underscoring its limited generalization to diverse content and tasks.

In contrast, the proposed adaptive luma range optimization demonstrates significant improvements in the BD-rate scores, ranging from -16.6% up to -28.0%. It corresponds to an improvement of up to nearly 5× over the prior method.

Fig. 3 presents the RD-curves of our adaptive method, the best performing configuration of the prior method, and the anchor on the OpenImages dataset for the OD task. The curves clearly show that our method consistently achieves the best coding efficiency across all data points, with particularly notable improvements in the low to mid BPP range.

Table 2 presents the class-wise average BD-rate results, along with the total averages, achieved by the proposed method on the SFU video dataset. For comparison, the table also includes the best-performing configuration of the prior constant luma scaling method

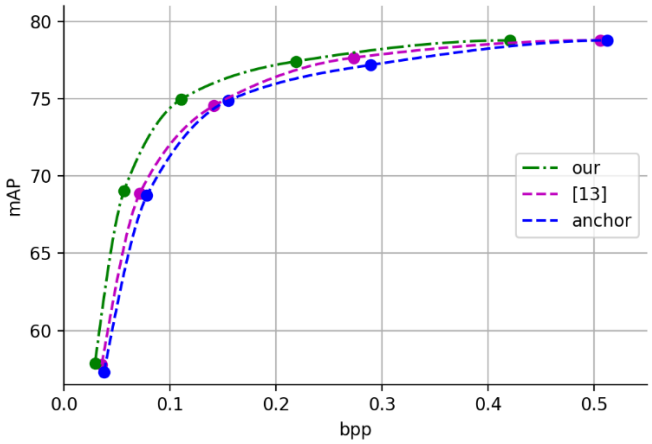

Fig. 3. RD-curves of the proposed method, the prior method [13], and the anchor on the OpenImages dataset for the OD task.

and its corresponding class-wise results, as reported in [13]. The total averages are computed as a weighted average across classes to account for the imbalance in the number of sequences per class.

This complementary test on the SFU demonstrates the *input analyzer's* ability to generalize to unseen resolutions, QPs, and uncompressed video sources, despite being exclusively trained on JPEG-compressed images from the OpenImages dataset, which exhibits significantly more limited resolution diversity than SFU. The SFU dataset comprises uncompressed videos sources with a wide range of resolutions, spanning from 240p to 1600p. The results demonstrate that the *input analyzer* maintains strong performance on the SFU dataset, achieving substantial BD-rate gains that, almost in all cases, surpass the result reported for the prior method. This is particularly noteworthy given that, in this experiment, the down-scaling factor was determined solely based on the analysis of the first frame of each sequence. Consequently, the observed improvements can be primarily attributed to the fully adaptive up-scaling prediction.

These results revealed that training the *input analyzer* on a wide range of QPs is not necessary to ensure its robustness across different QPs during inference time. In the final experiments, only three QP values (32, 37, 42) were used for training, which also accelerated the training process by reducing the amount of training data. Despite this limited range, the *input analyzer* demonstrated strong performance across all QPs in both the OpenImages and SFU evaluations. This is noteworthy given the high variability of QPs present in the SFU dataset.

No architectural optimization was performed for the *input analyzer* model in this work. The implemented model comprises approximately 858k parameters, and with the input resolution of 512×512, a single inference involves around 220 million FLOPs. While the architecture is already relatively lightweight, there remains potential for further optimization, which is left for future work.

6 CONCLUSION

In this paper, we proposed an adaptive luma range optimization method for VCM. In our approach, the input is analyzed prior to encoding to determine optimal luma down-scaling and up-scaling parameters, aiming to achieve the best trade-off between bitrate and machine task accuracy.

Table 2. BD-rate (%) results for the proposed method on SFU video dataset under AI, LD, and RA configurations

SFU OD	**Our method**			**Best average from [13]**		
	AI	LD	RA	AI	LD	RA
ClassA	-31.1	-45.4	+15.0	-15.6	-34.0	+5.2
ClassB	-12.2	-17.7	-17.7	-4.5	-13.8	-13.5
ClassC	-9.3	-14.3	-9.9	-5.4	-13.8	-6.5
ClassD	-7.9	-11.6	-16.9	-6.6	-12.8	-10.6
Average	**-11.4**	**-16.9**	**-12.5**	**-6.3**	**-15.0**	**-9.0**

The method was implemented using deep learning techniques, where a lightweight neural network serves as the analysis module. To facilitate its training, we introduced a novel training framework incorporating a codec proxy module, enabling end-to-end optimization within a conventional non-differentiable video codec. Evaluation results with VVC show that our adaptive method improved BD-rate scores substantially, up to nearly 5× compared to the improvements of the prior non-adaptive luma range scaling approach. Relative to the anchor, it achieves bitrate reductions of up to 28.0% on image datasets and up to 45.4% on video dataset.

Future works include exploring alternative architectures for the input analyzer and training experiments with more diverse datasets.

REFERENCES

[1] B. Bross *et al.*, "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3736-3764, Oct. 2021.

[2] G. J. Sullivan, J. -R. Ohm, W. -J. Han, and 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

[3] H. Choi and I. V. Bajić, "Scalable image coding for humans and machines," *IEEE Trans. Image Process*, vol. 31, pp. 2739-2754, Mar. 2022.

[4] ISO/IEC JTC 1/SC 29/WG 2, "Call for Proposals for Video Coding for Machines," MPEG Technical Requirements output document w21546, 138th MPEG meeting, April 2022. [Online] Available: https://www.mpeg.org/structure/technical-requirements.

[5] S. Liu, J. Chen, and J. Ström (ed.) "Optimization of encoders and receiving systems for machine analysis of coded video content (draft 8, for future updates)," document JVET-AK2030-v3, Geneva, CH, Jan. 2025. [Online] Available: https://www.jvet-experts.org/doc_end_user/documents/37_Geneva/wg11/JVET-AK2030-v3.zip.

[6] L. Duan, J. Liu, W. Yang, T. Huang, and W. Gao, "Video coding for machines: a paradigm of collaborative compression and intelligent analytics," *IEEE Trans. Image Process.*, vol. 29, pp. 8680-8695, Aug. 2020.

[7] S. Różek, O. Stankiewicz, S. Maćkowiak, and M. Domański, "Video coding for machines using object analysis and standard video codecs," in *Proc. Int. Conf. on Vis. Commun. and Image Process.*, Jeju, Republic of Korea, Dec. 2023.

[8] A. Zahra, M. Ghafoor, K. Munir, A. Ullah, and Z. Ul Abideen, "Application of region-based video surveillance in smart cities using deep learning," *Multimed Tools Appl*, vol. 83, no. 5, pp. 15313-15338, Dec. 2021.

[9] A. D. Bagdanov, M. Bertini, A. Del Bimbo, and L. Seidenari, "Adaptive video compression for video surveillance applications," in *Proc. IEEE Int. Symp. Multimedia*, Dana Point, CA, USA, 2011, pp. 190-197.

[10] A. Aliouat, N. Kouadria, M. Maimour, and S. Harize, "An efficient low complexity region-of-interest detection for video coding in wireless visual surveillance," in *Proc. Int. Multi-Conf. Syst., Signals & Devices*, Setif, Algeria, May 2022, pp. 1357-1362.

[11] A. Marie, K. Desnos, L. Morin, and L. Zhang, "Video coding for machines: large-scale evaluation of deep neural networks robustness to compression artifacts for semantic segmentation," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Shanghai, China, Sep. 2022.

[12] H. Choi, S. Jeong, S. Kwak, S. -H. Jung, and J. H. Ko, "Adaptive image downscaling for rate-accuracy-latency optimization of task-target image compression," in *Proc. Int. Conf. AI Circuits and Syst.,* Abu Dhabi, United Arab Emirates, Apr. 2024.

[13] T. Partanen *et al.*, "Luma range scaling for enhanced VVC efficiency in video coding for machines," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, West Lafayette, IN, USA, Oct. 2024.

[14] J. Liu, D. Liu, W. Yang, S. Xia, X. Zhang, and Y. Dai, "A comprehensive benchmark for single image compression artifact reduction," in *IEEE Trans. Image Process.*, vol. 29, pp. 7845-7860, July 2020.

[15] J. I. Ahonen, R. G. Youvalari, N. Le, H. Zhang, F. Cricri, and H. R. Tavakoli, "Learned Enhancement Filters for Image Coding for Machines," in *Proc. IEEE Int. Symp. Multimedia*, Naple, Italy, pp. 235-239, Nov. 2021.

[16] H. Choi and I. V. Bajic, "High efficiency compression for object detection," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, Calgary, Canada, Apr. 2018, pp. 1792-1796.

[17] Q. Cai, Z. Chen, D. O. Wu, S. Liu, and X. Li, "A novel video coding strategy in HEVC for object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4924-4937, Dec. 2021.

[18] K. Fischer, F. Fleckenstein, C. Herglotz, and A. Kaup, "Saliency-driven versatile video coding for neural object detection," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, Toronto, Canada, Jun. 2021, pp. 1505–1509.

[19] G. Lu, X. Ge, T. Zhong, Q. Hu, and J. Geng, "Preprocessing enhanced image compression for machine vision," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 12, pp. 13556-13568, Dec. 2024

[20] J. Terven, D.-M. Cordova-Esparza, J.-A. Romero-González, A. Ramírez-Pedraza, and E. A. Chávez-Urbiola, "A comprehensive survey of loss functions and metrics in deep learning," *Artif. Intell. Rev.*, vol. 58, no. 195, Apr. 2025.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation." 2015 [Online]. Available: https://arxiv.org/abs/1505.04597

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vision Pattern Recognit.*, Las Vegas, NV, USA, pp. 770-778, June 2016.

[23] C. Gao, D. Liu, L. Li, and F. Wu, "Towards task-generic image compression: a study of semantics-oriented metrics," in *IEEE Trans. Multimedia*, vol. 25, pp. 721-735, 2023.

[24] A. Paszke *et al.*, "Pytorch: an imperative style high-performance deep learning library," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2019, pp 8024–8035.

[25] "VVC Reference Software Version 20.0," Accessed: May. 2025. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-20.0.

[26] A.Kuznetsova *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale,", *Int. J. Comput. Vis.*, vol. 128, pp. 1956-1981, Mar. 2020.

[27] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, vol. 11218, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland : Springer, pp. 122-138, 2018.

[28] Ross Wightman, "PyTorch Image Models," 2019. Accessed: May 2025. [Online]. Available: https://github.com/rwightman/pytorch-image-models.

[29] S. Liu and C. Hollman, "Common test conditions for optimization of encoders and receiving systems for machine analysis of coded video content," *document JVET-AI2031-v1*, Sapporo, Japan, Jul. 2024.

[30] G. Bjøntegaard, "Improvements of the BD-PSNR model," document VCEG-AI11, Berlin, Germany, Jul. 2008.

[31] W. Gao, X. Xu, M. Qin, and S. Liu, "An open dataset for video coding for machines standardization," in *Proc. IEEE Int. Conf. Image Process.*, Bordeaux, France, Oct. 2022, pp. 4008-4012.

[32] Y. Wu, A. Kirillov, F. Masa, W.-Y. Lo, and R. Girschick, "Detectron2," 2019. Accessed: May 2025. [Online]. Available: https://github.com/facebookresearch/detectron2.

[33] H. Choi, E. Hosseini, S. Ranjbar Alvar, R. A. Cohen, and I. V. Bajić, "A dataset of labelled objects on raw video sequences," Data in Brief, vol. 34, 2021.