Adapting JPEG XS gains and priorities to tasks and contents

Benoit Brummer
intoPIX
Mont-Saint-Guibert, Belgium
b.brummer@intopix.com

Christophe de Vleeschouwer
Université catholique de Louvain
Louvain-la-Neuve, Belgium
christophe.devleeschouwer@uclouvain.be

Abstract

Most current research in the domain of image compression focuses solely on achieving state of the art compression ratio, but that is not always usable in today's workflow due to the constraints on computing resources.

Constant market requirements for a low-complexity image codec have led to the recent development and standardization of a lightweight image codec named JPEG XS.

In this work we show that JPEG XS compression can be adapted to a specific given task and content, such as preserving visual quality on desktop content or maintaining high accuracy in neural network segmentation tasks, by optimizing its gain and priority parameters using the covariance matrix adaptation evolution strategy.

1. Introduction

JPEG has been the most widely used image codec since its introduction in 1992 and more powerful standards such as JPEG2000 have failed to take up consequential market shares due in large part to their added complexity. The JPEG XS codec was standardized in 2019 (ISO/IEC 21122) [1] in response to market demand for lightweight image compression [2]. Its typical use-cases are to replace uncompressed data flow, whose bandwidth requirement is no longer affordable due to the ever increasing content resolution, and embedded systems, which are bound by the complexity of their integrated circuits and often by their battery capacity. Use-cases often target specific tasks and contents, hence it can be beneficial to optimize an image encoder to account for prior knowledge of its intended application.

Similar optimization work has been produced with other codecs; mainly JPEG, for which quantization tables have been optimized for different tasks using various methods [3][4][5][6], and JPEG2000, which has had its large parameter space explored with the use of genetic algorithms [7].

Akin to the JPEG quantization table [8], JPEG XS uses a table of gains and priorities for each sub-band. The ISO 21122-1 standard provides a table of gains and priorities (jointly referred to as weights) that maximize peak signal-to-noise ratio (PSNR) [1, Tab. H.3] and states that other values may result in higher quality for certain scenarios. These weights are embedded in the encoded image; the decoder parses them and the choice of optimized weights cannot break compatibility with ISO 21122 compliant decoders.

We describe a framework based on the covariance matrix adaptation evolution strategy (CMA-ES) to optimize JPEG XS weights for different metrics and contents. Through this method, decreases of up to 14% of bits-per-pixel (bpp) at constant quality are achieved on desktop content and up to 59% at constant accuracy on a semantic segmentation task.

2. Background

2.1. JPEG XS

The JPEG XS coding scheme [1] first applies a reversible color conversion from the RGB color space to $\mathrm{YC}_b\mathrm{C}_r$. The signal is then decomposed into a set of sub-bands, through an asymmetric 2D multilevel discrete wavelet transformation (DWT) [9]. Wavelet coefficients are split into precincts containing coefficients from all sub-bands that make up a spatial region of the image (usually a number of horizontal lines), and are encoded as 16-bit integers prior to quantization. The gains (G_b) and priorities (P_b) table makes up the weights, a pair of values defined for each sub-band (b), which are stored in the encoded picture header. The precinct quantization (Q_p) and precinct refinement (R_p) are computed for each precinct (p) based on the bit budget available for that precinct. The truncation position ($T_{\mathrm{b,p}}$) determines how many least significant bits are discarded from all wavelet coefficients of a given sub-band within a precinct; $T_{\mathrm{b,p}}$ is calculated from G_b, P_b, Q_p, and R_p as follow: $T_{\mathrm{p,b}} = Q_\mathrm{p} - \mathrm{G}_\mathrm{b} + r$ where r is 1 for $\mathrm{P}_\mathrm{b} < R_\mathrm{p}$ and 0 otherwise. Eventually $T_{\mathrm{p,b}}$ is then clamped to the valid range, i.e. [0;15]. Bitplane counts (i.e. number of non-zero bitplanes in a group of four coefficients) are then entropy-coded and packed along with the truncated coefficients values, their signs, and Q_p and R_p values.

2.2. CMA-ES

CMA-ES is an evolutionary algorithm which performs derivative-free numerical optimization of non-linear and non-convex functions, using a population of candidate solutions whose mean and covariance matrix are updated for each generation using the most fit individuals. [10][11]

Rios and Sahinidis [12] have extensively reviewed 22 derivative-free optimization algorithms over a wide range of problems. They show that CMA-ES outperforms other algorithms by $\sim$200% when measuring the fraction of non-convex non-smooth problems with 10-30 variables solved from a near-optimal solution [12, Figure 32]. The optimization of JPEG XS High profile weights fits this use-case, given there are 30 variables whose initial values have already been optimized to maximize PSNR, and the number of function evaluations is of little importance since this optimization is performed only once. The pycma library [13] provides a well-tested implementation of CMA-ES.

3. Experiments

The optimization method described in 3.1 introduces the CMA-ES optimizer and its pycma implementation, details the use of a JPEG XS codec, and lists the datasets and metrics used. Results are given in 3.2.

3.1. Method

CMA-ES implementation

We use the pycma [13] CMA-ES implementation. The only required parameters are an initial solution X_0 and the initial standard deviation σ_0. The gains defined in [1, Tab. H.3] are used for X_0 and their standard deviation is computed for σ_0. Population size is set to 14 by pycma based on the number of variables. Gains are optimized as floating-point values; their truncated integer representation is used as the gains given to the encoder and the fractional parts are ranked in descending order which is used as priorities. This matches the relative importance of priorities in Section 2.1, where a bit of precision is added to bands whose priority $\mathrm{P_b}$ is smaller than the precinct refinement threshold R_p.

JPEG XS implementation

JPEG XS High profile [2, Tab. 2] is optimized using TICO-XSM, the JPEG XS reference software [14] provided by intoPIX [15]. Each image is encoded and decoded with a given set of weights and the average loss (1-MS-SSIM, -PSNR, or the prediction error) on the image set is used as the fitness value. A single image is encoded and decoded using the `tco_enc_dec` command with a given target bpp and a configuration file containing the candidate weights. The gains and priorities are listed in the configuration file in the following order (deepest level first): $\mathrm{LL}_{5,2}$, $\mathrm{HL}_{5,2}$, $\mathrm{HL}_{4,2}$, $\mathrm{HL}_{3,2}$, $\mathrm{HL}_{2,2}$, $\mathrm{LH}_{2,2}$, $\mathrm{HH}_{2,2}$, $\mathrm{HL}_{1,1}$, $\mathrm{LH}_{1,1}$, $\mathrm{HH}_{1,1}$ where H denotes high-pass filtering and L is low-pass filtering. The list is repeated for each component (Y, C_b, C_r). This differs from [1, Tab. H.3], where the three components are alternating for each value.

Human visual system optimization

The training data used to generate weights optimized for image quality is a random subset of 240 featured pictures from Wikimedia Commons [16], randomly scaled between 0.25 and 1.00 of their original size and cropped to match the Kodak Image Dataset [17] resolution of 768x512. A different subset of 240 featured pictures and the 24-pictures Kodak Image Dataset are used for testing. A set of 200 screenshots has been collected on Wikimedia Commons for synthetic desktop content optimization, ensuring variability in the software, operating systems, fonts, and tasks displayed. The desktop content has a resolution of 1920x1080 and is split into two 100-picture subsets for training and testing. These images are optimized for MS-SSIM and PSNR.

Computer vision optimization

Optimized weights are generated for an AI-based computer vision task to minimize prediction error incurred from compression artifacts. This employs the Cityscapes dataset, consisting of street scenes captured with an automotive camera in fifty different cities with pixel-level annotations covering thirty classes. [18] The data is split between a "train" set used to train a convolutional neural network (CNN) that performs pixel-level semantic labeling on uncompressed content, and a "val" set used to test the accuracy of the trained CNN using the intersection-over-union (IoU $= \frac{\mathrm{TP}}{\mathrm{TP+FP+FN}}$) metric. The HarDNet architecture is well suited for this optimization task because it performs fast inference without forgoing analysis accuracy [19], enabling faster (and thus more) parameter evaluations in CMA-ES.

The "train" data cannot be used to evaluate IoU performance because the model has already seen it during training; therefore, we optimize the 500-image "val" data. This is split into three folds: two cities used as training data and one city used as testing data. The weighted average over three folds is based on the number of images in each test set. The MS-SSIM metric is also optimized on the Cityscapes dataset; the "train" and "train_extra" data can be reused, making two 100-images subsets for training and testing.

Similarly to the human visual system (HVS) targeted optimization, AI optimization is performed by encoding and decoding the whole set of images to be evaluated, computing the average IoU metric, and providing its result to the optimizer as the fitness value of the current set of weights.

Hardware and complexity

The weights optimization process is parallelized using all available CPU threads, each encoding an image. Image analyses such as semantic segmentation and MS-SSIM evaluation are computed faster on a GPU. Two or more optimization jobs are run in parallel, making constant use of all available computing resources. 27 weight configurations are extensively optimized and results are obtained in approximately three weeks from a total of 78 CPU threads.

Each training process is repeated at 1.00, 3.00, and 5.00 bpp. 4000 function evaluations are performed for each visual optimization and 1500 for each AI task. In addition, the AI task optimization is repeated over three folds and different bitrates are interpolated to produce Figure 2. The interpolation is carried out by computing the fitness value of the top-10 weights of the closest optimized bitrate at each interpolated bitrate, as well as performing a smaller 150 function evaluation optimization at the interpolated bitrates.

3.2. Results

Human visual system results

Results of the HVS weights optimization are summarized in Table 1. Weights have been optimized for different metrics and contents and tested with the MS-SSIM and PSNR metrics. Table 1 shows that MS-SSIM is consistently improved through the use of optimized weights, even when using different content classes. Improvement is especially considerable at low bitrate, as the standard weights consume 12% to 18% more bitrate to achieve the MS-SSIM index obtained with 1.00 bpp using optimized weights. These results are close to those obtained using the "visual" weights provided in the reference software. Improvements obtained with PSNR weights are negligible; the standard weights are already optimized for this metric and content-specific optimizations bring no significant improvement. Weights optimized for an AI task (IoU on Cityscapes) perform poorly when measured against PSNR or MS-SSIM, even on the same dataset. Figure 1 shows a visual comparison with weights optimized for the MS-SSIM metric on "Featured Pictures" and "Desktop" content at 1.00 bpp. MS-SSIM weights tend to yield a consistent level of detail for a globally acceptable visual quality at low bitrate, whereas PSNR-optimized weights tend to show a mix of sharper areas and bleeding artifacts.

AI results

Table 2 shows the performance improvements from weights optimized on a semantic segmentation task on the Cityscapes dataset, as well as content-specific MS-SSIM weights. The fully optimized weights achieve a 33.3% to 59% reduction in required bpp, meanwhile the MS-SSIM weights do not translate to performance gains on the IoU metric. Figure 2 shows the gains over many more bitrates, by testing the standard weights every 0.50 bpp and interpolating the optimized weights. The IoU score obtained on uncompressed content is 0.7506. This number is sometimes exceeded when performing inference on compressed content; the highest scores obtained are 0.7514 with optimized weights and 0.7508 with standard weights at 7.00 bpp.

Our results indicate that JPEG XS compression at a high enough bitrate does not appear to be detrimental to the performance of a semantic segmentation model trained on uncompressed content (starting at 4.25 bpp with optimized weights, or 6.90 bpp with standard weights), and it may even be beneficial in some instances, as the model performed better at 7.00 bpp than with uncompressed content.

The metric used for fitness evaluation appears to be much more effective than the type of data provided. This is seen with synthetic "Desktop" content optimized weights providing an MS-SSIM index on natural content close to that obtained with the weights optimized on featured pictures, PSNR weights bringing little to no benefit regardless of the content, and the MS-SSIM and IoU optimized weights not benefiting each other on the same content. Moreover, optimizing for different bitrates appears to be beneficial as different sets of weights are generated for each optimized bitrate.

4. Conclusion

CMA-ES is a black-box optimization method that can be used to optimize JPEG XS quantization parameters for a given task and content type. Such evolutionary algorithms work well with the small number of quantization parameters present in JPEG XS and CMA-ES provides a nearly parameter-free method for task-specific optimization of JPEG XS image compression.

A relatively small training set is used. This is necessary because each image needs to be encoded and decoded to test the given weights and scoring must remain consistent for each evaluation. Overfitting does not appear to be an issue even when the training and test sets are small and distinct, likely because the small number of weights do not provide the capacity to overfit.

PSNR weights optimized with CMA-ES do not vary significantly from those defined in the ISO standard, which were calculated to achieve maximum PSNR, regardless of the content type and bitrate used. MS-SSIM optimized weights are more beneficial; a bitrate reduction of 0.12 to 0.32 bpp (3.8% to 18%) is observed between 1.00 and 5.00 bpp (depending on the type of content) and the gain in MS-SSIM index translates to a higher perceptual quality.

The weights optimized for an AI-analysis task show the versatility of this method, as they adapt to a very specific task and content and provide a more notable reduction in required bitrate at constant accuracy (33.3% to 59% at the optimized bitrates). The fitness function can be tuned to fit any objective function. For example, one could combine the MS-SSIM index and an AI metric to create hybrid human-machine weights.

Table 1. Results obtained with weights optimized for different contents and visual metrics (optimization shown in the left two columns). The test metrics, content, and bpp are shown on top. The best weight for each test is marked as such in bold. The last two columns show the bpp percentage increase needed by standard (or visual [14]) weights to match the score of the optimized (underlined) weights.

		MS-SSIM									PSNR			
		FeaturedPictures			kodak			screenshots			kodak	screenshots		
weights dataset	weight metric	1.00 bpp	3.00 bpp	5.00 bpp	1.00 bpp	3.00 bpp	5.00 bpp	1.00 bpp	3.00 bpp	5.00 bpp	3.00 bpp	1.00 bpp	3.00 bpp	5.00 bpp
Cityscapes	IoU	0.96002	0.98799	0.99240	0.95979	0.99299	0.99713	0.98344	0.99822	0.99946	38.451	30.337	50.302	59.631
	MS-SSIM	0.96631	0.99121	0.99440	0.96711	0.99574	0.99888	0.98780	0.99897	**0.99974**	39.183	31.541	50.976	62.450
Desktop	MS-SSIM	0.96632	0.99128	0.99441	0.96644	0.99571	0.99887	**0.99216**	0.99898	**0.99974**	39.573	36.393	52.289	62.612
	PSNR	0.95965	0.99026	0.99406	0.95907	0.99516	0.99869	0.98942	0.99869	0.99967	40.959	**38.228**	**53.584**	**64.124**
FeaturedPictures	MS-SSIM	**0.96699**	**0.99133**	**0.99442**	**0.96717**	**0.99580**	**0.99889**	0.99176	**0.99906**	0.99973	39.772	35.932	52.273	63.091
	PSNR	0.96133	0.99037	0.99415	0.96098	0.99516	0.99876	0.98971	0.99886	0.99970	**41.014**	37.725	53.482	64.087
JPEG-XSM visual	Human	0.96547	0.99079	0.99416	0.96569	0.99556	0.99877	0.99137	0.99891	0.99970	39.324	35.937	50.477	60.651
ISO 21122 std.	PSNR	0.96060	0.99038	0.99416	0.96009	0.99522	0.99876	0.98957	0.99885	0.99970	40.999	38.147	53.569	64.085
% bpp improvement over visual weights		4.00	6.00	6.80	3.00	2.77	3.40	6.00	7.00	5.00	18.7	23.0	19.7	15.8
% bpp improvement over std. weights		14.0	9.67	6.4	12.0	5.33	3.80	18.0	5.33	4.00	3.01	1.00	0.33	0.2

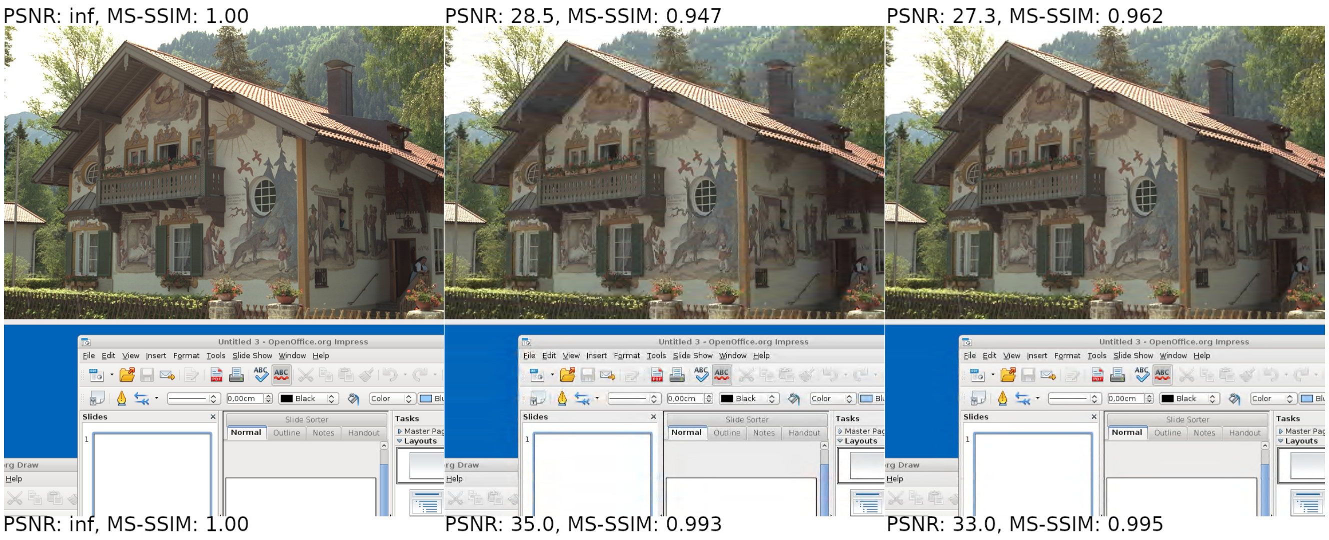
PSNR: inf, MS-SSIM: 1.00
PSNR: 28.5, MS-SSIM: 0.947
PSNR: 27.3, MS-SSIM: 0.962
PSNR: inf, MS-SSIM: 1.00
PSNR: 35.0, MS-SSIM: 0.993
PSNR: 33.0, MS-SSIM: 0.995

Figure 1. Visual comparison of "Featured Pictures" and "Desktop" MS-SSIM weights at 1.00 bpp. Left: uncompressed image, middle: compressed with ISO 21122 PSNR weights, right: compressed with MS-SSIM optimized weights (ours), top: kodak image 24, bottom: Tails 0.12 office screenshot.

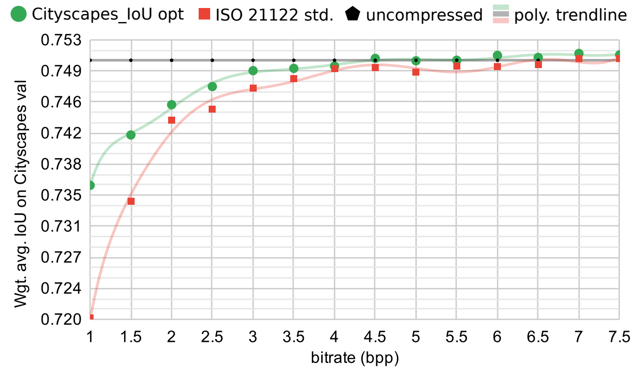

Figure 2. IoU/bitrate performance on Cityscape validation set. JPEG XS standard weights compared to optimized weights (weighted average over three folds). Optimization is performed at 1.00, 3.00, and 5.00 bpp, other bitrates are interpolated as described in section 3.1.

This search for optimized weights is by no means exhaustive as there are more use-cases to optimize for than could possibly be listed in this work. Even so, it presents a simple and effective method to optimize JPEG XS quantization parameters for any specific task. Furthermore, this method may be used to optimize JPEG XS image compression for AI image analysis models which would themselves be prohibitively expensive (or impossible) to fine-tune for compressed content.

Table 2. Average IoU obtained over 3 folds using weights optimized for semantic segmentation on Cityscapes. Also shown are weights optimized for MS-SSIM on Cityscapes, and the MS-SSIM index obtained at 3.00 bpp (rightmost column).

		IoU			MS-SSIM
weight dataset	weight metric	1.00 bpp	3.00 bpp	5.00 bpp	3.00 bpp
Cityscapes	IoU	**0.73583**	**0.74937**	**0.75054**	0.99791
	MS-SSIM	0.72882	0.74707	0.74971	**0.99876**
JPEG-XSM visual	Human	0.73164	0.74742	0.74990	0.99862
ISO 21122 std.	PSNR	0.72018	0.74731	0.74920	0.99857
% bpp improvement over std. weights		59.0	33.3	34.8	10.0

5. Acknowledgements

I would like to thank Antonin Descampe for his help in formalizing the JPEG XS background section, and Pascal Pellegrin for helping me understand the JPEG XS codec and analyzing the resulting weights.

References

[1] ISO, *Information technology — JPEG XS low-latency lightweight image coding system — Part 1: Core coding system*, vol. 2019. May 2019. 1, 2

[2] A. Descampe, J. Keinert, T. Richter, S. Fößel, and G. Rouvroy, "Jpeg xs, a new standard for visually lossless low-latency lightweight image compression," in *Applications of Digital Image Processing XL* (A. G. Tescher, ed.), vol. 10396, pp. 68 – 79, International Society for Optics and Photonics, SPIE, 2017. 1, 2

[3] Y. Jiang and M. S. Pattichis, "Jpeg image compression using quantization table optimization based on perceptual image quality assessment," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pp. 225–229, Nov 2011. 1

[4] M. Hopkins, M. Mitzenmacher, and S. Wagner-Carena, "Simulated annealing for jpeg quantization," in *2018 Data Compression Conference*, pp. 412–412, March 2018. 1

[5] J. Chao, H. Chen, and E. Steinbach, "On the design of a novel jpeg quantization table for improved feature detection performance," in *2013 IEEE International Conference on Image Processing*, pp. 1675–1679, Sep. 2013. 1

[6] E. Tuba, M. Tuba, D. Simian, and R. Jovanovic, "Jpeg quantization table optimization by guided fireworks algorithm," in *International Workshop on Combinatorial Image Analysis*, pp. 294–307, Springer, 2017. 1

[7] Yani Zhang, B. T. Pham, and M. P. Eckstein, "Automated optimization of jpeg 2000 encoder options based on model observer performance for detecting variable signals in x-ray coronary angiograms," *IEEE Transactions on Medical Imaging*, vol. 23, pp. 459–474, April 2004. 1

[8] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992. 1

[9] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, July 1989. 1

[10] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001. 2

[11] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016. 2

[12] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, pp. 1247–1293, 2013. 2

[13] N. Hansen, Y. Akimoto, and P. Baudis, "CMA-ES/pycma on Github." Zenodo, DOI:10.5281/zenodo.2559634, Feb. 2019. 2

[14] ISO, *Information technology — JPEG XS low-latency lightweight image coding system — Part 5: Reference software*, vol. 2020. Jan. 2020. 2, 4

[15] "intopix tico-xs — jpeg-xs ip-cores and sdks (iso/iec 21122)." `https://www.intopix.com/jpeg-xs`. Accessed: 2020-02-18. 2

[16] "Commons:featured pictures - wikimedia commons." `https://commons.wikimedia.org/wiki/Commons:Featured_pictures`. Accessed: 2020-02-19. 2

[17] "True color kodak images." `http://r0k.us/graphics/kodak/`. Accessed: 2020-01-20. 2

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[19] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin, "Hardnet: A low memory traffic network," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2