PATCH-BASED IMAGE LEARNED CODEC USING OVERLAPPING

Marwa Tarchouli[1,2], Marc Riviere[1], Thomas Guionnet[1], Wassim Hamidouche[2], Meriem Outtas[2], and Olivier Deforges[2]

[1]Ateme, Rennes, France
[2]Univ. Rennes, INSA Rennes, CNRS, IETR - Rennes, France

ABSTRACT

End-to-end learned image and video codecs, based on auto-encoder architecture, adapt naturally to image resolution, thanks to their convolutional aspect. However, while coding high resolution images, these codecs face hardware problems such as memory saturation. This paper proposes a patch-based image coding solution based on an end-to-end learned model, which aims to remedy to the hardware limitation while maintaining the same quality as full resolution image coding. Our method consists in coding overlapping patches of the image and reconstructing them into a decoded image using a weighting function. This approach manages to be on par with the performance of full resolution image coding using an endto-end learned model, and even slightly outperforms it, while being adaptable to different memory sizes. Moreover, this work undertakes a full study on the effect of the patch size on this solution's performance, and consequently determines the best patch resolution in terms of coding time and coding efficiency. Finally, the method introduced in this work is also compatible with any learned codec based on a conv/deconvolutional autoencoder architecture without having to retrain the model.

KEYWORDS

Auto-encoders, Image compression, Deblocking, block artifacts.

1. INTRODUCTION

During the past few years, deep learning based end-to-end image and video coding field achieved great improvements, and performance of such algorithms can now compete with traditional coding systems like JPEG [1], JPEG2000 [2] or BPG [3].

Their auto-encoder architecture, built with convolutional layers, enables processing different image resolutions, no matter the resolution used during the training step. However, with growing model sizes or picture and video resolutions (4K, 8K), these solutions face hardware memory saturation. In fact, authors in [26] present a comprehensive study of computational limitations related to deep-learning models, and they rightly predict that the constraints will grow up.

For example, coding a standard resolution such as an HD image using a powerful GPU as NVIDIA GeForce RTX 2080ti with a memory capacity of 11Go, is not possible as it does not fit into the memory requirement. For 4K resolution, it is of course worse.

One way to solve this issue is to use a patch-based coding approach. The image is divided into patches having the same size that can be encoded independently. The patch size should be smaller than the image size and should fit into the memory constraints. Then, the decoded patches are gathered to reconstruct the decoded image, as illustrated in Fig.1. Moreover, this method enables to implement several forms of parallelization as well as the ability to access a sub-part of the image without entirely decoding it. Flexible image subdivision is classically required in practical video compression applications. The latest video coding standard, Versatile Video Coding (VVC) [27], provides three means for subdividing pictures, namely slices, tiles and sub-pictures. The latter can for instance be used for viewport access in virtual reality (VR) applications without decoding or even transmitting the whole bitstream. The two formers are traditionally used for processing parallelization.

The proposed solution addresses the hardware limitation issues, but the reconstructed picture can have block artifacts in the patch boundaries, widely deteriorating the image quality (Fig.2).

This paper is an extension of our previous work [28], the main goal is to address the hardware limitation issue of end-to-end learned codecs without deteriorating the decoded image quality (i.e. remove block artifacts). To do so, overlapping patches are encoded and then a weighting function is used to reconstruct the overlapping pixels. This method provides an objective and subjective quality slightly higher than the full image encoding approach, while leveraging the memory consumption flexibility of the patch-based coding solutions. However, it comes with a slight increase in coding time.

Then, in order to determine the patch size that provides the best outcome in terms of trade-off between coding time and performance, this paper also presents a study about the effect of the patch size on the coding efficiency.

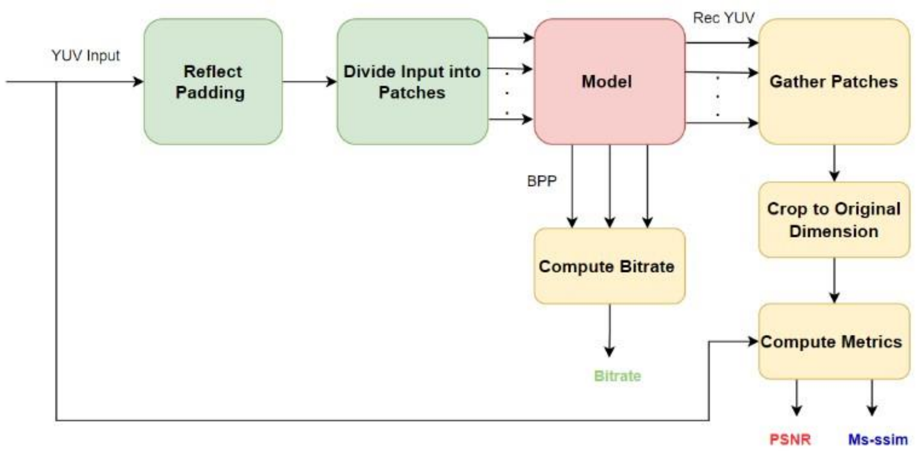


Fig.1 Steps to patch-based image coding

The rest of the paper is structured as follows. In section 2, the related work on end-to-end learned image coding as well as approaches to eliminate block artifacts are presented. Then, the patchbased image coding using overlapping method is explained in section 3. Section 4 presents the results of the driven experiments. Afterwards, the impact of the patch size on the proposed method is analysed. Finally, section 6 concludes the paper.

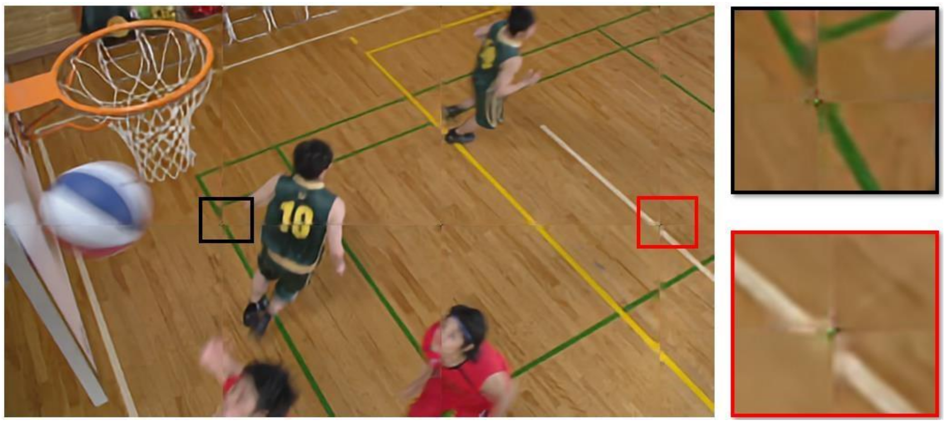


Fig.2 Reconstructed image per patch using an end-to-end learned codec

2. Related Work

2.1. End-to-End Learned Image Compression

Recently, deep learning codecs have accomplished promising results in a short period of time. In fact, end-to-end image compression managed to reach the state-of-the-art of traditional image coding in terms of coding efficiency. Authors in [4] proposed an image compression structure based on a conv/deconvolutional auto-encoder [5] whose role is to transform the input image into a latent representation. Then, a factorized entropy model is used to estimate the probability distribution of this representation allowing entropy coding. Ballé et al. [6] replaced the factorized entropy model by a hyperprior auto-encoder. This mechanism enables the probability model to adapt itself to the input image as well as to capture spatial dependencies in the latent representation.

This work is considered as the reference design for numerous other state-of-the-art contributions. In the same context, [7] uses an autoregressive module to improve the entropy model. It succeeds in improving the coding performance compared to [6]. However, this improvement leads to an increase in complexity. [8] presented a model architecture based on residual blocks [9] and attention modules in order to extract a compact and an efficient latent representation. In addition, it presented an entropy model which exploits Gaussian mixture model to parameterize the latent distributions. This latter method achieves performances that compete with the versatile video coding (VVC) [10] encoder in Intra mode.

Some works [11] used the ability of generative adversarial network (GAN) [12] to generate realistic and sharp images for image compression at low bitrate and for small resolutions, while [13], [14] combined adversarial loss with rate-distortion loss to include the fidelity aspect into the network allowing better subjective quality for high resolution images.

Recurrent neural networks (RNN) are used in some approaches [15], [16]. They present progressive models with architectures designed on top of residuals. These approaches achieve results better than JPEG. However, they perform worse than conv/deconvolutional auto-encoder solutions with entropy models like [8], while having a higher complexity.

The auto-encoder solution presented in [8] achieves the best performance among learned image codecs. It is competitive with VVC, the latest traditional codec. Despite the performance of these solutions, they require specific hardware to operate effectively (GPU), which are limited in memory capacity.

2.2. Deblocking with Neural Networks

Deep learning approaches were also explored for post-processing decoded images, particularly to eliminate block artifacts. Some works focused on enhancing quality of images coded by traditional codecs. For instance, [17] provides a network architecture to remove block artifacts from JPEG compressed blocks using neighbouring blocks, while [18] tempted to deblock the whole decoded image. [19] introduced pre & post-processing networks to improve JPEG compression performance. In fact, [19] proposed an end-to-end framework composed of two convolutional neural networks (CNN) along with a handcrafted image codec such as JPEG, JPEG2000 or BPG. The first CNN learns an optimal and a compact representation of the image, which is compressed by the traditional image codec. Then, the second CNN enhances the quality of the decoded image. The results of [19] outperform several state-of-the-art methods of image deblocking and denoising. However, this work was only tested on grayscale, images with small resolutions, which is not compatible with real compression applications.

Other works were interested in deblocking images coded by learned image codecs, especially progressive learned codecs. For instance, [20] proposed a patch-based image coding framework called BINet which uses binarized neighbouring patches to eliminate block artifacts. Each patch is reconstructed using 9 surrounding binarized neighbours. This approach outperforms JPEG at low bitrates.

[21] and [22] address the same problem by introducing a postprocessing network to remove blocking artifacts, which increases the size of the model and the complexity of the training process.

With a different objective, super resolution models, such as VDSR [23], showed promising results in enhancing decoded image quality.

Nevertheless, all these related works are either networks which require a dependence on traditional image codecs, or their training process is difficult, and retraining is necessary if the learned codec changes, or, they are not consistent with codecs based on conv/deconvolutional auto-encoders architectures such as [8]. This paper proposes a patch-based image coding approach to address the hardware limitation of any conv/deconvolutional learned image codec without requiring a retraining. It achieves the same quality level as the full resolution coding and enables a flexible use of memory which make the method undemanding in terms of hardware material. However, these results are achieved at the expense of a slight increase of coding time (about 3%).

3. PATCH-BASED IMAGE CODING USING OVERLAPPING

3.1. Proposed Method

In this approach, the process described in Fig.1 is followed. First, reflect padding is applied to the input image in order to make its size divisible by the patch size. While dividing the image into patches, every two consecutive patches must have a range of pixels in common horizontally and vertically, as it is illustrated in Fig.3. Then, the coding step is performed by an end-to-end learned image codec on input patches of size $P + N$, where P is the size of the patch w/o overlapping and N is the number of the overlapping pixels. In the next step, the image is reconstructed from the patches. The overlapping areas are combined by a weighting function which generates a progressive transition from one patch to the other. In fact, if b_m and $b_{m+1}$are two consecutive reconstructed patches overlapping horizontally on Npixels, the value of the i^{th} overlapping pixel $p_{rec}(i)$, for a given line in the reconstructed image is determined by the following equation:

$$p_{rec}(i) = \left(1 - \frac{i}{N-1}\right) p_{b_m}(P+i) + \left(\frac{i}{N-1}\right) p_{b_{m+1}}(i), \quad (1)$$

where $i \in \{0, \ \ldots \ , \ N-1\}$ is the index of the overlapping pixels, P is the size of the patch w/o overlapping, p_{bm} and p_{bm+1} are pixels values, for a specific line, of two consecutive decoded patches b_m and b_{m+1} respectively.

The same equation is valid for vertically overlapping patches. Once the decoded image is reconstructed, quality metrics can be computed.

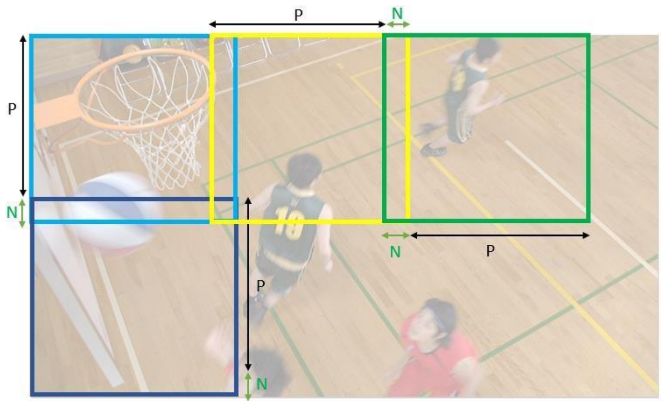


Fig.3 Method to overlap patches on N pixels

3.2. Training and Evaluation

The end-to-end learned codec used in this paper is an implementation of the model architecture introduced in [8]. It is a conv/deconvolutional auto-encoder model leveraging the efficiency of residual blocks and attention modules, in addition to using a Gaussian Mixture model as the entropy engine. It was trained on 400 000 samples from the CLIC 2020 dataset [24]. For training, 256x256 sized patches were randomly cropped from each image of the training set. The model was trained on a total of 500 000 steps. It was optimized using an Adam optimizer, with a batch size of 4. The learning rate value was set to 10^{-4} for the first 200 000 steps and then it was decreased to 10^{-5} for the rest of the training. The loss function to be minimized is the ratedistortion loss function formulated as :

$$J = D + \lambda R, \quad (2)$$

where D refers to the distortion between the original patch and the reconstructed one, measured by the Mean Square Error (MSE) or the Multi-Scale Structural Similarity Index [25] (MS-SSIM) metrics, and R refers to the rate used to transmit the bitstream, estimated using the Shannon entropy. λ is the Lagrangian multiplier, allowing to adapt the bit rate targeted by the learned image coding model. The goal is the teach the model to minimize the distortion between the original patch and the reconstructed one while using a reasonable bit rate.

Eight models have been trained, 4 for each quality metric (MSE and MS-SSIM), matching 4 different bit rates. The corresponding Lagrangian multipliers are λ = {420, 220, 120, 64} for MSSSIM models and λ= {4096, 3140, 2048, 1024} for MSE models.

Our method is then evaluated on Class B, C, D, E and F of the JVET Common Test Conditions (CTC) sequences (8 bit sequences), which have different resolutions (1920x1080, 1280x720, 840x832 and 416x240). The hardware limitation this paper addresses, start to occur at class B and class E resolutions, but it also concerns class A sequences. Although Class A is not included in this evaluation, the solution presented in this paper can be generalised to class A sequences.

For each sequence, one frame is extracted and compressed both entirely (referred as the full image approach) and by the proposed patch-based approach, with and without overlapping. We used N Є {0, 2, 4, 8, 16, 32} overlapping pixels and we set $P = 256$ similar to the training resolution.

4. EXPERIMENT RESULTS

4.1. Coding Time and Memory Analysis

Our method allows using the available GPU memory in a flexible way by coding multiple patches simultaneously. In fact, instead of coding the largest possible patches sequentially, we explored the GPU ability to parallelize processing. Therefore, the model was fed a batch of patches as input. The input shape becomes then [B, C, H, W] where C, H and W refer, respectively, to the channel number, the height, and the width of the input patch while B corresponds to the number of patches processed in parallel, called the batch size.

While fixing the patch size (W and H), the batch size value can be adapted to the available GPU memory. Hence, if the total number of patches to code cannot be fed to the model as one batch, the input patches are regrouped to smaller batches.

Table 1 compares the coding time of our proposed approach using parallelization, with the coding time of the full resolution learned coding. For information purposes, the coding time of two more methods is computed: our approach without parallelization and the patch-based learned image coding without overlapping using parallelization.

To achieve this experiment, images of different resolutions were extracted from the JVET CTC and were coded using two machines with two powerful GPUs: Nvidia GeForce RTX 2080ti and Nvidia GeForce RTX 3090 with memory capacity of 11Go and 24Go respectively.

Full resolution coding of an HD image is not possible on both GPUs due to Out Of Memory (OOM) error, while full coding an 1280x720 image, can only be run on the machine with the GPU RTX 3090 since it has more available memory (24Go). It is important to note that these resolutions are standard resolutions in practical applications of image compression. Therefore, the fact that they cannot be run on one of the latest GPUs is inconvenient. In this case, our method provides a solution that enables coding high resolution images without deteriorating quality.

While our method is necessary to code high resolution images (HD, 720p, ...), it increases coding time of the system for smaller resolutions. For instance, when running the resolution 832x480 on

2080ti GPU, patch-based coding increases the coding time by 3 % compared to full resolution coding. This is expected since our method requires coding more pixels in order to overlap patches. Patch-based coding w/o overlapping confirms this explanation since it has approximately the same coding time as full resolution coding.

Table 1: Coding Time

Resolution	Method	Coding time GPU 2080 11Go	Coding time GPU 3080 24Go	Total Number of patches	Number of patches coded in parallel	N
HD	Full resolution coding	OOM*	OOM*	-	-	-
	Patch coding in parallel w/o overlapping	3.18s	1.96s	40	8	-
	Patch coding in parallel with overlapping	3.39s	2.03s	40	8	16
	Patch coding sequentially with overlapping	4.39s	2.32s	40	-	16
1280x720	Full resolution coding	OOM*	0.93s	-	-	-
	Patch coding in parallel w/o overlapping	1.75s	0.95s	15	5	-
	Patch coding in parallel with overlapping	1.91s	1.01s	15	5	16
	Patch coding sequentially with overlapping	2.73s	1.25s	15	-	16
832x480	Full resolution coding	1.06s	0.52	-	-	-
	Patch coding in parallel w/o overlapping	1.05s	0.54s	8	8	-
	Patch coding in parallel with overlapping	1.10s	0.55s	8	8	16
	Patch coding sequentially with overlapping	1.586s	0.70s	8	-	16

** "OOM" stands for "Out Of Memory"*

To conclude this section, our approach addresses the hardware limitation problem since it allows coding resolutions such as HD and 720p. Moreover, it enables flexible adaptation to available memory. However, it increases slightly the coding time of the coding system. We believe this increase in coding time is worthwhile for the solution this method proposes, especially for coding high resolution images.

4.2. Rate-Distortion aVisual Results

Table 2 sums up the BD-rate gain of the patch-based learned image coding with and without overlapping comparing to learned full image coding on CTC sequences, using an end-to-end model trained to minimize MS-SSIM as distortion metric. For the purpose of this evaluation, full image approach for resolutions such as HD and 720p, was run on CPU. It is important to note that using CPU to run an end-to-end leaned codec is not considered because it is severely time consuming.

Patch-based image coding without overlapping and with 2 overlapping pixels (N = 2) presents a loss in BD-rate, comparing to full image coding. The first observation confirms the block artifacts issue caused by patch-based approaches. The second one is explained by the fact that two overlapping pixels are not enough to recover from these artifacts. Yet, the BD-rate loss of our proposed method with N=2 is decreased significantly compared to patch-based coding without overlapping. As the number of overlapping pixels increases, the loss in BD-rate decreases which indicates that the block artifacts are removed efficiently. Hence, our method has managed to be on par with the performance of full image coding.

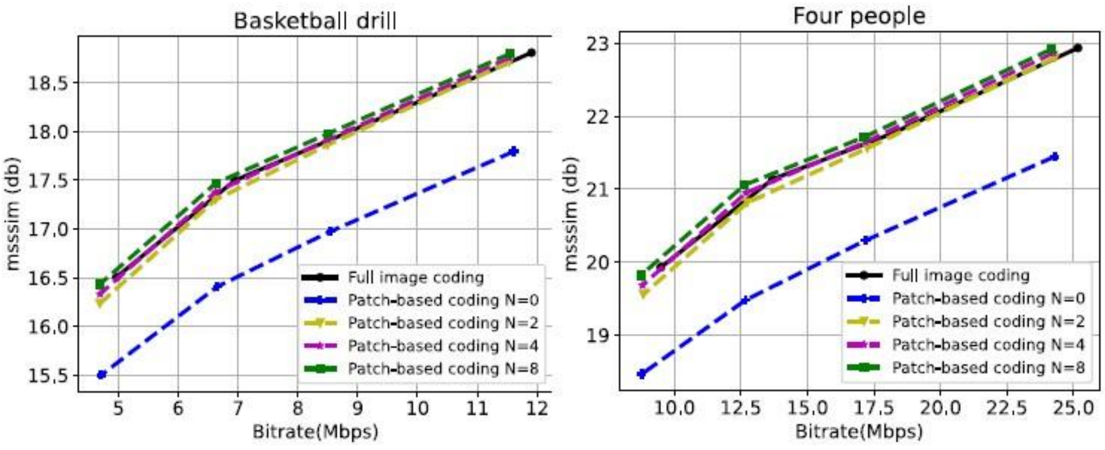
In fact, with N = 8 marginal gains are observed compared to full image coding. Though, these gains are at their peak with 16 overlapping pixels. As the end-to-end coding model was trained on 256x256 cropped images, it performs better on coding patches of similar size than coding the full resolution image, which explains the gain in BD-rate. For N > 16, BD-rate gains saturation is observed.

Table 2: BD-rate (MS-SSIM) gains (%) of patch-based coding schemes compared to full image coding system for CTC sequences.

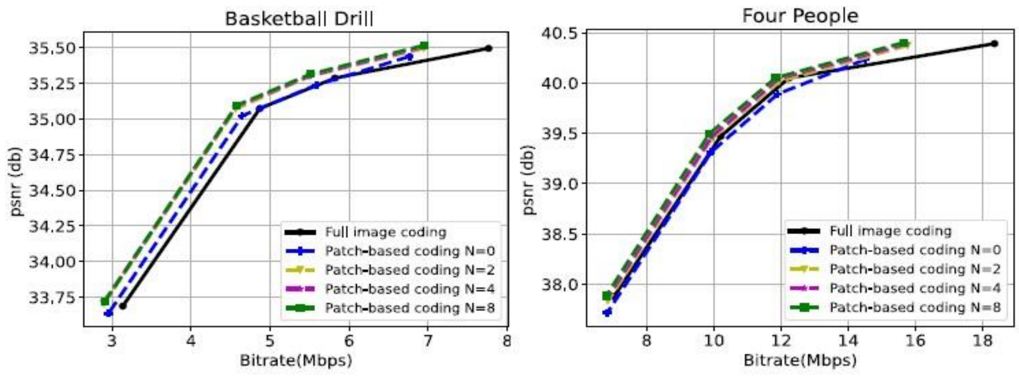
	Sequence	Patch w/o Overlapping	Patch with Overlapping				
			N = 2	N = 4	N = 8	N = 16	N = 32
Class B	Cactus	0.573	-0.0003	-0.033	-0.066	-0.083	-0.079
	BasketballDrive	0.630	0.062	0.024	-0.013	-0.032	-0.030
	BQTerrace	0.744	0.025	-0.020	-0.060	-0.083	-0.090
Class C	RaceHorses	0.503	0.013	-0.014	-0.041	-0.052	-0.048
	BasketballDrill	0.484	0.022	-0.004	-0.034	-0.049	-0.051
	BQMall	0.732	0.056	0.014	-0.023	-0.044	-0.038
	PartyScene	0.428	0.032	0.005	-0.020	-0.032	-0.027
Class D	BasketballPass	0.286	0.025	0.010	-0.012	-0.015	-0.009
	BlowingBubbles	0.146	0.019	0.006	-0.008	-0.016	-0.012
	BQSquare	0.214	0.020	0.008	-0.005	-0.011	-0.012
	RaceHorses	0.281	0.026	0.009	-0.011	-0.022	-0.022
Class E	Johnny	1.240	0.081	0.041	-0.004	-0.026	-0.029
	FourPeople	0.643	0.030	-0.012	-0.046	-0.068	-0.072
	KristenAndSara	0.991	0.062	0.021	-0.015	-0.043	-0.043
Class F	BasketballDrillText	0.506	0.035	0.010	-0.019	-0.035	-0.035
	SlideShow	1.018	0.081	0.028	-0.006	-0.031	-0.031
	SlideEditing	0.533	0.019	-0.006	-0.030	-0.035	-0.031

Rate-distortion curves for MS-SSIM models are reported in Fig.4(a) and confirm previous observations. The patch-based coding approach with overlapping allows to fill the gap with the full image coding one. While reaching a better quality, the proposed method increases slightly the rate comparing to patch-based coding without overlapping, which is expected as more pixels are encoded.

The decoded images of the methods mentioned before are visualized in Fig.5. Overlapping with N = 2 (Fig.5.d) and N = 4 (Fig.5.e) reduces the block artifacts while overlapping with 8 pixels (Fig.5.f) eliminates them entirely.



(a) MS-SSIM Rate-Distortion results



(b) PSNR Rate-Distortion results

Fig.4 Rate-Distortion results of the proposed approach compared to Full Image coding and patch-based image coding without overlapping for Basketball Drill and Four People sequences.

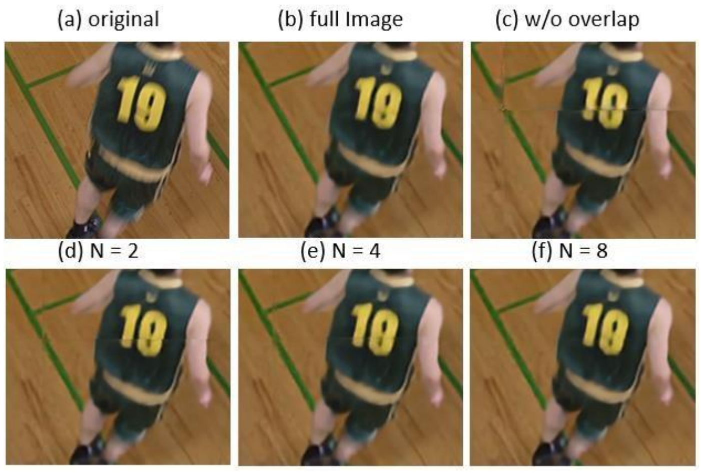


Fig.5 Visual results for Basketball Drill using MS-SSIM model with λ = 420

The BD-rate results with distortion measured with Peak Signal to Noise Ratio (PSNR) are provided in Table 3, for the end-to-end learned codec trained to optimize the MSE metric. Compared with full image coding, patch-based image coding without overlapping showed a BDrate loss for PSNR metric (Table 3) less important than the BD-rate loss for MS-SSIM metric (Table 2). This is also illustrated in Fig.4(b) as the quality gap between image coding per patch without overlapping and the full image coding has narrowed. Furthermore, although block artifacts exist in the decoded image (Fig.6.c), they are less visible than the decoded image resulting from the MS-SSIM end-to-end codec (Fig.5.c). This behaviour may be explained by the fact that MS-SSIM is computed using a sliding window while MSE is a pixel-based metric.

In any case, the proposed approach improves the BD-rate performance, the rate-distortion curves (Fig.4) and the perceptual quality of the decoded image (Fig.6 and Fig.5).

Table 3: BD-rate (PSNR) gains (%) of patch-based coding schemes compared to full image coding system for CTC sequences.

	Sequence	Patch w/o Overlapping	Patch with Overlapping				
			N = 2	N = 4	N = 8	N = 16	N = 32
Class B	Cactus	-0.022	-0.073	-0.080	-0.086	-0.096	-0.089
	BasketballDrive	0.042	-0.010	-0.010	-0.010	-0.013	-0.010
	BQTerrace	-0.040	-0.100	-0.113	-0.121	-0.132	-0.123
Class C	RaceHorses	0.010	-0.033	-0.037	-0.045	-0.046	-0.040
	BasketballDrill	-0.030	-0.071	-0.077	-0.078	-0.079	-0.070
	BQMall	0.013	-0.018	-0.023	-0.020	-0.021	-0.010
	PartyScene	0.033	-0.001	-0.006	-0.010	-0.016	-0.020
Class D	BasketballPass	-0.014	-0.047	-0.046	-0.050	-0.040	-0.022
	BlowingBubbles	0.005	0.001	0.002	-0.008	-0.014	-0.012
	BQSquare	0.017	0.008	0.007	0.014	0.015	0.020
	RaceHorses	0.009	-0.005	-0.010	-0.013	-0.017	-0.017
Class E	Johnny	0.049	0.016	-0.002	-0.013	-0.023	-0.010
	FourPeople	0.016	-0.023	-0.031	-0.045	-0.070	-0.070
	KristenAndSara	0.042	-0.001	-0.011	-0.026	-0.058	-0.030
Class F	BasketballDrillText	0.011	-0.031	-0.036	-0.040	-0.042	-0.040
	SlideShow	0.051	-0.030	-0.032	-0.026	-0.034	-0.020
	SlideEditing	0.029	-0.006	-0.009	-0.016	-0.016	-0.010

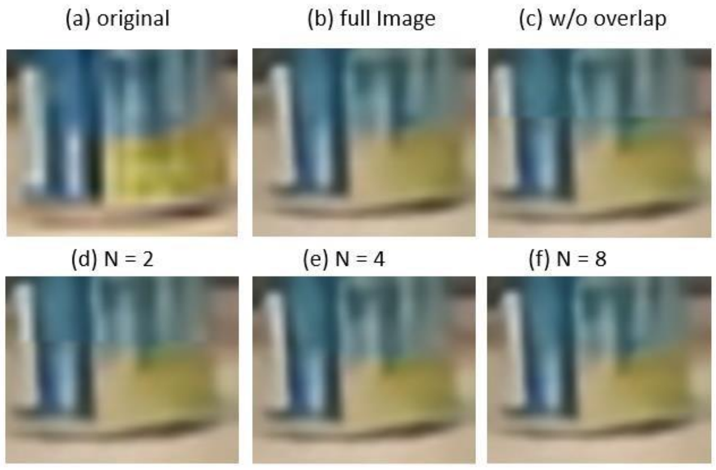


Fig.6 Visual results for Four People using MSE model with λ = 4096

5. PATCH SIZE IMPACT ON PATCH BASED CODING WITH OVERLAPPING

In this section, we study the impact of the patch size on the performance of our method. Hence, we aim to determine the patch size which provide the best results especially in terms of coding time and memory consumption.

For this purpose, our approach is evaluated with different patch sizes, with and without parallelization, on Class B sequences of the JVET CTC which has the HD resolution. This resolution allows exploring a bigger range of patch sizes. For instance, testing our method with a patch resolution of 768x768, is possible on an HD frame while it is not possible with a frame which resolution is 1280x720.

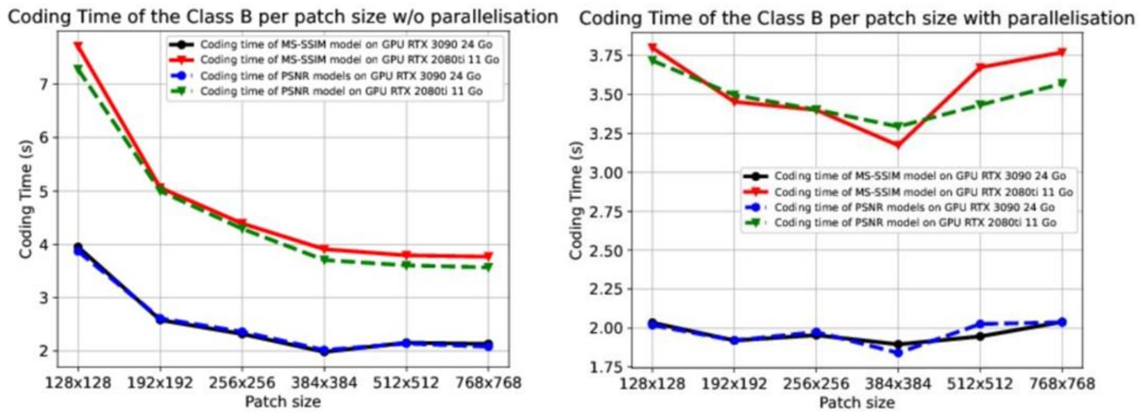
Furthermore, according to the first results presented in the previous section, the number of overlapping pixels that gives the best results in terms of coding efficiency, equals to 16. Thus, we set N = 16 in this study.

5.1. Square Patch Sizes

5.1.1. Coding Time and Memory Analysis

First, a range of square patch size is explored: P Є {128x128, 192x192, 256x256, 384x384, 512x512, 768x768}. Then, in Table 4, the coding time on both GPUs : RTX 2080ti with 11Go and RTX 3090 with 24Go, is measured with and without parallelization. We tested different values of the number of patches processed in parallel going from 1, where parallelization is not used, to the maximum value that is possible giving the amount of memory available on the GPU. For example, for the patch size 128x128, 30 is the maximum number of patches that can be coded in parallel if run on the GPU "RTX 2080ti with 11Go". Beyond this value, the memory saturation problem occurs.

Table 4 compares the coding time of our approach with and without parallelization for each patch size. For a given a patch size, the coding time decrease while increasing the number of the patches coded in parallel. This shows the importance of parallelization in reducing the coding time and taking advantage as much as possible from the possible memory available.



(a) Without parallelization (b) With parallelization

Fig.7 Impact of the patch size on the coding time with and w/o parallelization.

Fig.7 shows how the execution time of our method, run on both GPUs, varies with different patch sizes. Without parallelization (Fig.7.a), our method has the highest coding time when the patch size is the smallest. Then it decreases when increasing the patch size. This is because when the frame is divided into small patches, the total number of patches is high. Each patch is transmitted separately to the GPU and then processed by that GPU sequentially, while with parallelization, a set of patches are transmitted at once, then the GPU process those patches simultaneously. The transmission of one single patch takes less time than the transmission time of one set of patches. However, coding the whole frame sequentially is slower.

In fact, Fig.7.b illustrates the results of the execution time, according to different patch sizes, while using parallelization. The number of patches coded in parallel is chosen so that the maximum number of patches, allowed in the GPU memory, are processed simultaneously. As a result, the coding time is reduced significantly for small patch size such as : 128x128, 192x192, 256x256 and 384x384 while for the 512x512 and 768x768, parallelization is less efficient. This can be explained by the fact that only 1 or 2 patches can be coded simultaneously for these patch sizes.

In these cases, there might still have available memory in the GPU but not enough to process another patch. Hence, we do not benefit from all the available memory. This assumption is confirmed in Table 5 where we measure the memory consumption of our approach using different patch sizes on the GPU RTX 2080ti with the memory capacity of 11Go. For 512x512 and 768x768 patch sizes, only 8 Go out of 11Go is used where for other patch sizes such as 128x128 and 192x192, the consumed memory is up to 10 Go. To sum up, high patch size does not allow an optimal memory consumption and does not benefit from parallelization to reduce the execution time.

However, small patch size should not be privileged either since they come along with more overlapping pixels to code and thus increasing the coding time. Therefore, the patch size that is considered a trade-off between optimal memory consumption and coding added overlapping pixels, is 384x384.

Table 4: Coding time of the patch based coding with overlapping using different patch sizes with and w/o parallelization.

	128x128					192x19					256x25				
	N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go		N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go		N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go	
		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)
Class B	135	1	7,71	1	3,95	60	1	5,06	1	2,58	40	1	4,39	1	2,32
	135	15	4,00	27	2,15	60	8	3,76	12	1,99	40	4	3,71	8	2,03
	135	27	3,89	33	2,09	60	12	3,67	30	1,96	40	8	3,39	10	1,97
	135	30	3,79	71	2,03	60	15	3,45	36	1,92	40	-	-	20	1,94

	384x384					512x51					768x76				
	N Total patches	11 GO		24 GO		N Total patches	11 GO		24 GO		N Total patches	11 GO		24 GO	
		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)
Class B	15	1	3,90	1	1,98	12	1	3,79	1	2,15	6	1	3,76	1	2,13
	15	2	3,29	3	1,96	12	2	3,67	2	2,11	6	-	-	2	2,03
	15	3	3,17	5	1,91	12	-	-	3	2,01	-	-	-	-	-
	15	4	3,17	10	1,89	12	-	-	6	1,94	-	-	-	-	-

Table 5: Memory Consumption per patch size . The Used GPU is RTX 2080ti with the memory capacity of 11 Go

Patch Size	GPU memory consumption
128x128	10784MiB / 11016MiB
192x192	10496MiB / 11016MiB
256x256	9302MiB / 11016MiB
384x384	9470MiB / 11016MiB
512x512	8306MiB / 11016MiB
768x768	8864MiB / 11016MiB

5.1.2. Rate-Distortion and Visual Results

Here, the coding efficiency of our approach is evaluated on class B sequences using different patch sizes. For this experiment, the number of overlapping pixels is fixed to N = 16, since according to the previous section this value provides the best results in terms of rate-distortion results. Then, the BD-rate of our method compared to full coding system using MS-SSIM and MSE models are computed for different patch sizes in Table 6 and Table 7 respectively.

Table 6: BD-rate (MS-SSIM) gains (%) of patch-based coding with overlapping (N = 16) compared to full image coding system using different patch size for Class B sequences.

	Sequence	Patch size					
		128x128	192x192	256x256	384x384	512x512	768x768
Class B	BasketballDrive	-0,095	-0,077	-0,080	-0,049	-0,046	-0,045
	BQTerrace	-0,067	-0,052	-0,030	-0,018	0,004	-0,0001
	Cactus	-0,096	-0,072	-0,080	-0,038	-0,060	-0,062
Average		-0,086	-0,067	-0,064	-0,035	-0,034	-0,035

Table 7: BD-rate (PSNR) gains (%) of patch-based coding with overlapping compared to full image coding system using different patch size for Class B sequences.

	Sequence	Patch size					
		128x128	192x192	256x256	384x384	512x512	768x768
Class B	BasketballDrive	-0,104	-0,105	-0,125	-0,076	-0,082	-0,097
	BQTerrace	-0,014	-0,022	-0,012	0,004	0,027	-0,001
	Cactus	-0,064	-0,057	-0,093	-0,037	-0,073	-0,076
Average		-0,060	-0,061	-0,076	-0,036	-0,042	-0,058

As expected, slight gains are obtained for almost all patch sizes, for both MS-SSIM and MSE models. Moreover, Fig.8 and Fig.9 represent the visual results for our solution using two different patch sizes for MSE and MS-SSIM models. In all cases, border artifacts are successfully eliminated no matter the patch size is.

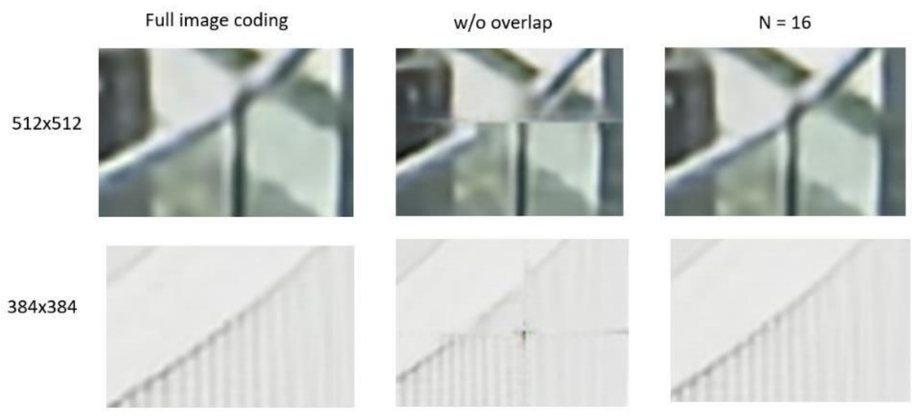


Fig.8 Visual results for BQTerrace using MS-SSIM model with λ = 420, for patch size : 384x384 and 512x512

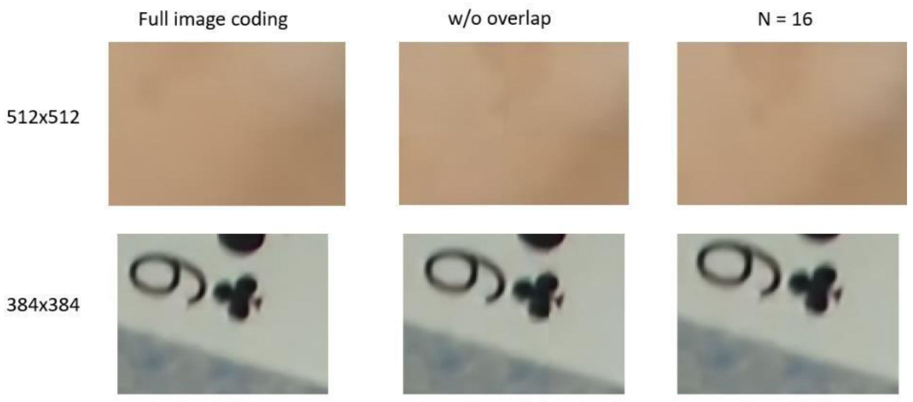


Fig.9 Visual results for Cactus using MSE model with λ = 4096, for patch size: 384x384 and 512x512

Next, the behaviour of the BD-rate gains according to patch size is visualised in Fig.10. The BDrate used is the BD-rate average among all Class B sequences. The learned codec used is trained to optimize MS-SSIM as quality metric. This choice is explained by the fact is MS-SSIM is a metric computed using a sliding window making it aware of neighbour pixels, while MSE is a pixel wise metric. As a result, it does not take into consideration, border artifact.

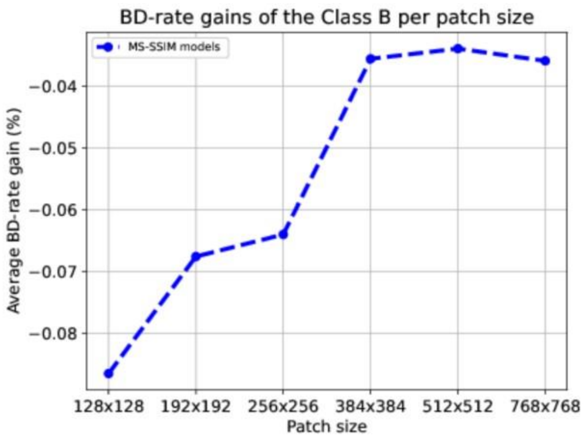


Fig.10 Variation of the BD-rate gains as per patch size

Fig.10 shows that BD-rate gains increase when the patch size is small. This behaviour confirms that learned codecs perform better in low resolution patches since they were trained on small image crops. Nevertheless, the more patch size decreases, the more overlapping pixels are added to be coded. Therefore, at some point, the overhead of the overlapping pixel will lead to deteriorate the performance of our method in terms of coding efficiency.

Hence, for square patch size, we privilege the patch resolution 384x384 because it allows parallelization and benefits largely (Table 5) from the available memory, which reduce the coding time. Moreover, it provides a BD-rate gain, without risking deteriorating the rate-distortion results due to the overlapping pixels overhead.

5.2. Patch Sizes Adapted to Practical Applications

5.2.1. Coding time and memory analysis

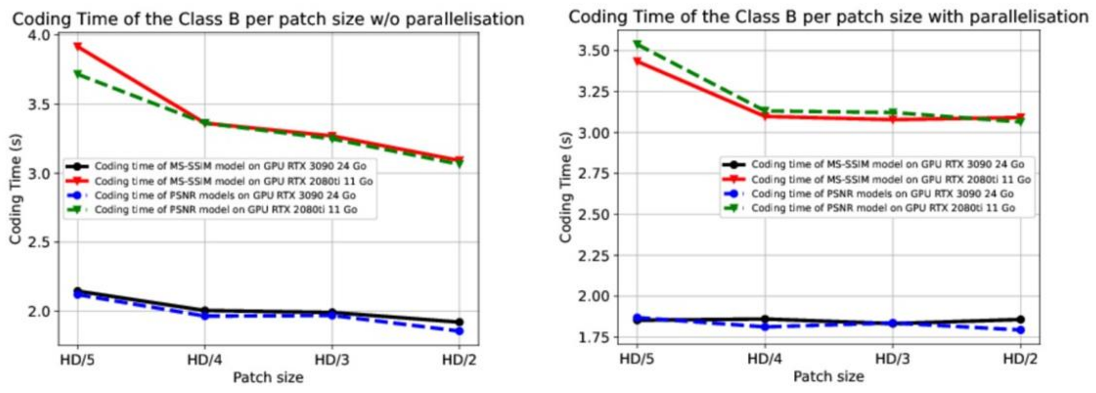
In this part, we test commonly used patch resolutions that are close to HD/2, HD/3, HD/4 and HD/5, which corresponds respectively to 960x540, 640x360, 480x272, 384x216.

Table 8 presents the results of our approach run on the same two GPUs mentioned previously, while using parallelization. For all patch sizes, the coding time decreases when the number of patches coded in parallel increases.

Table 8: Coding time of the patch based coding with overlapping using different patch sizes with and w/o parallelization. Range of patch sizes : HD/2, HD/3, HD/4 and HD/5

	HD/5					HD/4				
	N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go		N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go	
		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)
Class B	25	1	3,91	1	2,14	16	1	3,36	1	2,00
	25	3	3,63	10	1,86	16	2	3,23	4	1,89
	25	5	3,51	15	1,86	16	4	3,09	8	1,88
	25	7	3,43	17	1,85	16	5	3,09	12	1,86

	HD/3					HD/2				
	N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go		N Total patches	GPU RTX 2080ti 11Go		GPU RTX 3080 24 Go	
		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)		Nbr of patches coded in parallel	Coding Time (s)	Nbr of patches coded in parallel	Coding Time (s)
Class B	9	1	3,26	1	1,99	4	1	3,09	1	1,92
	9	2	3,11	3	1,88	4	-	-	2	1,92
	9	3	3,07	6	1,85	4	-	-	-	
	9	-	-	7	1,83	4	-	-	-	-



(a) Without parallelization (b) With parallelization

Fig.11 Impact of the patch size on the coding time with and w/o parallelization. The range of patch sizes is : HD/2, HD/3, HD/4 and HD/5

To enhance the benefits of parallelization, the curve of the variation of the execution time according to the patch size with and without parallelization is illustrated in Fig.11 . As before, the execution time of our method without parallelization is the highest when the patch size had the lowest resolution (HD/5). Then, the use of parallelization reduces the coding time significantly for the patch sizes: HD/5, HD/4 and HD/3, while it does not improve the coding time of the proposed approach with HD/2 as a patch size. In fact, Table 9 shows that the high patch size of HD/2 does not benefit completely from the memory capacity available.

Table 9: Memory Consumption per patch size. The Used GPU is RTX 2080ti with memory capacity of 11 Go and the range of patch sizes is : HD/2, HD/3, HD/4 and HD/5

Patch Size	GPU memory consumption
HD/5	10250MiB / 11016MiB
HD/4	10162MiB / 11016MiB
HD/3	10546MiB / 11016MiB
HD/2	8024MiB / 11016MiB

However, in Fig.11.b, the coding time with parallelization for HD/5 patch size, run on the GPU “RTX 2080ti 11Go”, is still higher comparing to the other patch sizes. This might be because of the added overlapping pixels.

We aim to determine a patch size that allows benefiting as much as possible from the available memory in the GPU and at the same time avoids a significant overhead of the overlapping pixels. As a compromise to these both constraints, we privilege the patch size HD/4 (480x272).

5.2.2. Rate-Distortion and Visual Results

Afterwards, in Table 10 and Table 11, the BD-rate gains are computed to compare the proposed solution with the full coding image with MSE and MS-SSIM models, for the patch sizes: HD/2 (960x540), HD/3 (640x360), HD/4 (480x272), HD/5 (384x216). In general, our method manages to achieve slight gains for both MSE and MS-SSIM models, for all patch sizes. Fig.12 and Fig.13 show the visual results of our method with the resolutions HD/4 and HD/5, for MSE and MSSSIM models. In any case, the border artifacts are entirely eliminated.

Table 10: BD-rate (MS-SSIM) gains (%) of patch-based coding with overlapping compared to full image coding system using different patch size for Class B sequences. The range of patch sizes is HD/2, HD/3, HD/4 and HD/5

	Sequence	Patch size			
		HD/5	HD/4	HD/3	HD/2
Class B	BasketballDrive	-0,039	-0.025	-0,025	-0,015
	BQTerrace	-0,04	-0.017	-0,016	-0,006
	Cactus	-0,043	-0.026	-0,026	-0,012
Average		-0,041	-0,023	-0,022	-0,011

Table 11: BD-rate (PSNR) gains (%) of patch-based coding with overlapping compared to full image coding system using different patch size for Class B sequences. The range of patch sizes is HD/2, HD/3, HD/4 and HD/5

	Sequence	Patch size			
		HD/5	HD/4	HD/3	HD/2
Class B	BasketballDrive	-0,045	-0,041	-0,036	-0,033
	BQTerrace	-0,032	-0,010	-0,010	-0,004
	Cactus	-0,027	-0,023	-0,021	-0,011
Average		-0,034	-0,025	-0,023	-0,016

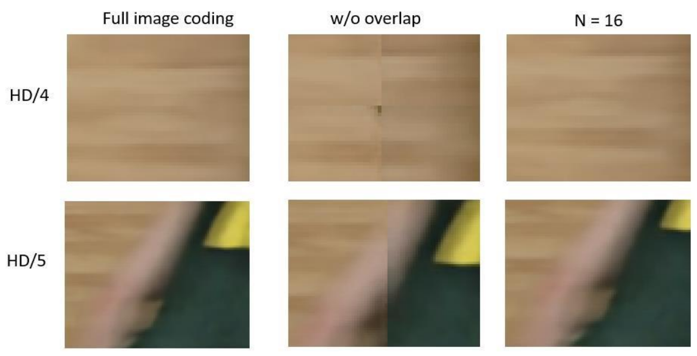


Fig.12 Visual results for BasketballDrive using MS-SSIM model with λ = 420, for patch size : HD/4 and HD/5

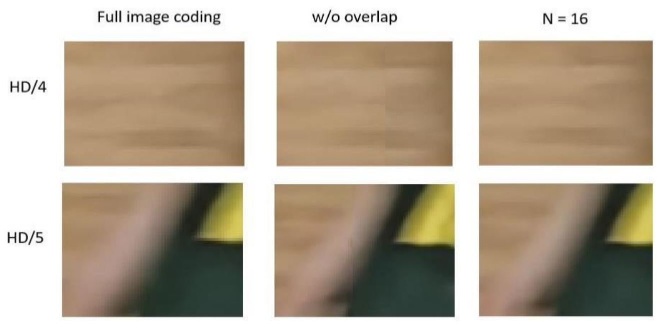


Fig.13 Visual results for BasketballDrive using MSE model with λ = 4096, for patch sizes : HD/4 and HD/5

Fig.14 illustrates the variation of BD-rate gains per patch size. The behaviour is similar to Fig.10, the BD-rate gains increases when the patches are smaller. Nevertheless, using low patch resolutions with our approach leads an important overhead of overlapping pixels. Therefore, small patch sizes cannot be privileged.

In terms of coding efficiency, our method with the patch size HD/4 allows achieving slight BDrate gains (0.23%) comparing to full image coding, as well as avoiding a significant overhead of additional overlapping.

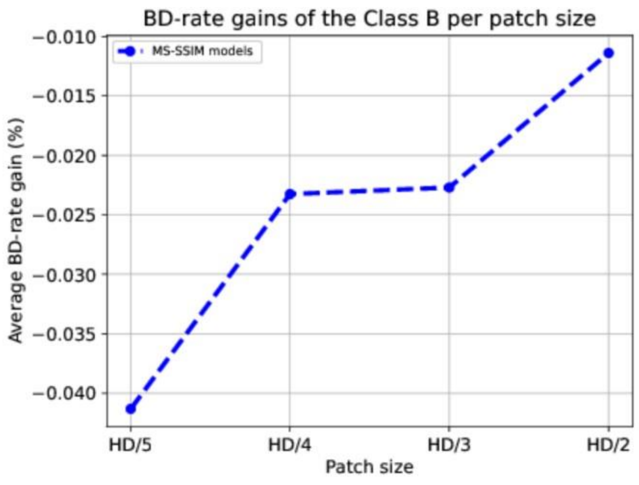


Fig.14 Variation of BD-rate gains as per to patch resolution. Patch resolutions : HD/5, HD/4, HD/3 and HD/2

To conclude this section, the impact of the patch size on our approach was analysed using two different ranges of patch sizes. In terms of coding time, the bigger the patch size is the less the parallelization is effective. In addition, the proposed method does not benefit fully from the available memory in case of high patch resolution, which increases the coding time of the solution. However, small patch size cannot be privileged either since they increase the overhead of the overlapping pixels. Therefore, the patch size that present a trade-off between both constraints corresponds in the first range of patch sizes to 384x384 and in the second range to 480x272 (HD/4). It is important to note that both these patch resolutions are close in their total number of pixels, which shows that our results are consistent.

In terms of coding efficiency, the patch size does not have an important impact on the performance of our method. It manages to achieve marginal gains compared to the full image approach. Nevertheless, it is expected that the use of small patch sizes will lead to BD-rate loss at some point because of the overhead of overlapping pixels.

6. CONCLUSION AND PERSPECTIVES

This work proposes a solution to the memory saturation problem that end-to-end learned codecs face while coding high resolution images such as HD. This solution consists in patch-based image coding while removing block artifacts using overlapping. This method benefits from flexible memory consumption and manage not only to achieve full image coding performance, but also to improve it slightly (-0.034% for MSE models and -0.024% for MS-SSIM models), even though it increases lightly the coding time (about 3%). Results are provided on JVET CTC sequences with MSE and MS-SSIM based models. The network architecture of [8] is used as our baseline end-to-end learned codec, but this method is compatible with any learned codec based on a conv/deconvolutional auto-encoder architecture.

Furthermore, this work provides an analysis on the effect of the patch size on the performance of the proposed solution. Thereby, the best patch in terms of coding time and coding efficiency is determined. In this study, Class B sequences were targeted since they have a high resolution (HD) which make them more exposed to the hardware limitation this paper aims to address. Moreover, two different ranges of patch sizes were explored. The number of overlapping pixels was set to 16.
In our future works, we aim to remedy to the increase in coding time, caused by coding additional overlapping pixels, by training an end-to-end image coding model to smooth the patch borders and hence remove the border artifacts automatically.

REFERENCES

[1] G. K. Wallace (1991) “The JPEG Still Picture Compression Standard,”, IEEE Transaction on Consumer Electronict, pp. 1—17

[2] C. Christopoulos & A. Skodras & T. Ebrahimi (2000) “The jpeg2000 still image coding system: an overview”, vol. 46, pp. 1103-1127.

[3] F. Bellard, “BPG Image format https://bellard.org/bpg/”.

[4] J.Ballé & V. Laparra & E. P Simoncelli (2017) “End-to-end optimized image compression” ICLR 2017 - Conference Track Proceeding.

[5] "P. Vincent & H. Larochelle & Y. Bengio & P.-A. Manzagol (2008) “Extracting and composing robust features with denoising autoencoders”, Intl. conf. on Machine Learning (ICML) .

[6] J. Ballé & D. Minnen & S. Singh & S.J Hwan, N.Johnston (2018) “Variational image compression with a scale hyperprior” ICLR 2018 - Conference Track Proceedings.

[7] J. Ballé & D. Minnen & G.Toderici (2018) “Joint Autoregressive and Hierarchical Priors for Learned Image Compression” Advances in Neural Information Processing Systems, pp. 1077110780.

[8] Z. Cheng & H. Sun & M. Takeuchi & J. Katto (2020) "Learned image compression with discretized gaussian mixture likelihoods and attention modules" IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7936-7945.

[9] K. He & X. Zhang & S. Ren & J. Sun (2016) “Deep Residual Learning for Image Recognition” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778.

[10] G. J. Sullivan & J. R. Ohm (2018) “Versatile video coding Towards the next generation of video compression” Picture Coding Symposium, vol. Jun-2018.

[11] S. Santurkar & D. Budden & N. Shavit (2018) “Generative Compression” Picture Coding Symposium (PCS), pp. 258-262.

[12] I. Goodfellow & J. Pouget-Abadie & M. Mirza & B. Xu & D. Warde-Farley & S. Ozair & A. Courville & Y. Bengio (2014) “Generative adversarial nets” Advances in Neural Information Processing Systems, vol. 3, pp. 2672-2680.

[13] F. Mentzer & G. Toderici & M. Tschannen & E. Agustsson (2020) “High-fidelity generative image compression” Advances in Neural Information Processing Systems, vol. 33, pp. 11913-11924.

[14] E. Agustsson & M. Tschannen & F. Mentzer & R. Timofte & L. Van Gool (2019) “Generative Adversarial Networks for Extreme Learned Image Compression” IEEE/CVF International Conference on Computer Vision (ICCV), pp. 221-231.

[15] G. Toderici & D. Vincent & N. Johnston & S. Hwang & D. Minnen & J. Shor & M. Covell (2017) “Full resolution image compression with recurrent neural networks” CVPR 2017, vol. 2017, pp. 5435-5443.

[16] G. Toderici & S. M. O’Malley & S. Hwang & D. Vincent &D. Minnen & S. Baluja & M. Covell & R. Sukthankar (2016) “Variable rate image compression with recurrent neural networks” ICLR - Conference Track Proceedings.

[17] D. Maleki & S. Nadalian & M. M. Derakhshani & M. A. Sadeghi (2018) “BlockCNN: A Deep Network for Artifact Removal and Image Compression” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

[18] L. Ke & B. Bahetiyaer & Y. Bo (2017) “An Efficient Deep Convolutional Neural Networks Model for Compressed Image Deblocking” Conference, IEEE International, pp. 1320--1325.

[19] F. Jiang & W. Tao & S. Liu & J. Ren & X. Guo & D. Zhao (2018) “An End-to-End Compression Framework Based on Convolutional Neural Networks” IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, pp. 3007--3018.

[20] A. Nortje & W. Brink & H. Engelbrecht & H. Kamper (2021) “BINet: A binary inpainting network for deep patch-based image compression” Signal Processing: Image Communication, vol. 92.

[21] W. Yaojun & L. Xin. & Z. Zhizheng& J. Xin & C. Zhibo (2022) “Learned Block-Based Hybrid Image Compression” IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, pp. 3978-3990.

[22] Z. Zhenghui. & J. Chuanmin. & W. Shanshe & M. Siwei & Y. Jiansheng (2021) “Learned Image Compression Using Adaptive Block-Wise Encoding and Reconstruction Network” IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5.

[23] J. Kim. & J. Lee. & K. Lee (2016) “Accurate image super-resolution using very deep convolutional networks” CVPR, pp. 1646--1654.

[24] “Workshop and C. on Learned Image Compression” Https://www.compression.cc/.

[25] Z. Wang & E. P. Simoncelli & A. C. Bovi (2003), "Multiscale structural similarity for image quality assessment," The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, Vol.2, pp. 1398-1402

[26] N. C. Thompson, K. Greenewald, K. Lee, & G. F. Manso, (2020). The computational limits of deep learning. arXiv preprint arXiv:2007.05558.

[27] 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. 37363764, Oct. 2021, doi: 10.1109/TCSVT.2021.3101953

[28] M. Tarchouli, S. Pelurson, T. Guionnet, W. Hamidouche, M. Outtas & Olivier Deforges (2022), "Patch-Based Image Coding with End-To-End Learned Codec using Overlapping", 12th International Conference on Artificial Intelligence, Soft Computing and Applications (AIAA 2022), vol 12, num 23