Learned Lossy Image Compression for Volumetric Medical Data

Jan Kotera[1,2,*], Matthias Wödlinger[1] and Manuel Keglevic[1]

[1]CVL, TU Wien, Favoritenstraße 9/11, 1040 Vienna, Austria
[2]Institute of information theory, CAS, Pod Vodárenskou věží 4, 182 00 Prague, Czech Republic

Abstract
This work addresses the problem of lossy compression of volumetric images consisting of individual slices such as those produced by CT scans and MRI machines in medical imaging. We propose an extension of a single-image lossy compression method with an autoregressive context module to a sequential encoding of the volumetric slices. In particular, we remove the intra-slice autoregressive relation and instead condition the entropy model of the latent on the previous slice in the sequence. This modification alleviates the typical disadvantages of autoregressive contexts and leads to a significant increase in performance compared to encoding each slice independently. We test the proposed method on a dataset of diverse CT scan images in a setting with an emphasis on high-fidelity reconstruction required in medical imaging and show that it compares favorably against several established state-of-the-art codecs in both performance and runtime.

Keywords
Learned Image Compression, Medical Image Data, Deep Learning

1. Introduction

Medical imaging is a set of techniques and processes that produce images of the interior of the body for the purpose of clinical analysis, medical intervention, or visual representation of the function of the internal organs. Examples of common types of imaging systems are x-rays, computed tomography (CT) scans, magnetic resonance imaging (MRI), or ultrasound (US). Medical imaging has become a staple tool not only for medical diagnosis and treatment but also a crucial component of research, as it allows researchers and physicians to establish a knowledge base of normal anatomy and physiology to make it possible to identify abnormalities and study the effects of medical intervention. For these reasons, the amount of image data produced in healthcare and medical research is huge and increasing [1], as are the requirements for efficient transmission and especially storage.

Image compression methods are designed for exactly that – to enable more efficient coding of image data with little or no loss in visual quality. The first successful image compression techniques were developed in the early 1990s and some of those are still being widely used today, such as for example the well-known JPEG method [2]. In recent years the development of novel compression methods for image and video accelerated, in line with the growing amount of streamed image and video data. Modern image compression codecs such as BPG [3], AVIF [4], or WebP [5] typically appear as by-products of a video codec development – the intra-frame component is extracted from the video codec and used as a standalone image codec.

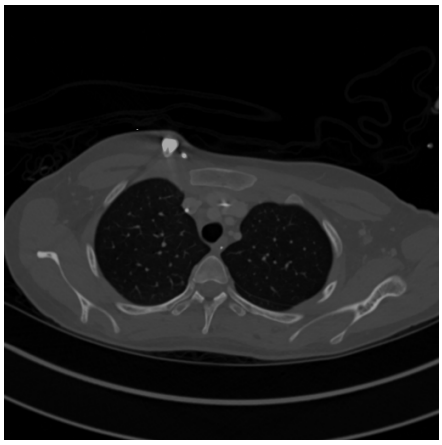

Figure 1: Illustrative example of a single uncompressed slice from the CT scan test set [6] used for performance evaluation.

For mainstream everyday use in applications such as image or video streaming, video calls, online gaming and so on the goal is for the reconstructed image to appear "natural and artefact-free" on first glance while achieving high enough compression ratios to make the above mentioned applications feasible. General-purpose video codes are therefore developed for and tested mainly on natural sequences, screen content, or synthetic scenes (eg. [7]) and typically benchmarked in perceptually lossy

26th Computer Vision Winter Workshop, Robert Sablatnig and Florian Kleber (eds.), Krems, Lower Austria, Austria, Feb. 15-17, 2023
*Corresponding author.
kotera@utia.cas.cz (J. Kotera); mwoedlinger@cvl.tuwien.ac.at (M. Wödlinger); keglevic@cvl.tuwien.ac.at (M. Keglevic)

range of < 40dB reconstruction PSNR (eg. [8]). similarly for image codecs. In the case of medical imaging the fundamental requirement is that the reconstruction error must not alter the subsequent clinical analysis. The reconstructed image must remain true to the original up to imperceptible "noise" void of any structure. We argue that using an established and straightforward objective metric such as PSNR for measuring the reconstruction error is the right approach here to ensure that the reconstructed image is truly nearly identical to the original when the reconstruction error is near zero. In our subjective tests (on HDR display) we find that we are not able to distinguish between the original and reconstructed images above 55dB PSNR, so that is approximately our target quality range. On the other hand, bellow 50dB we could identify loss of subtle structure in some images. Having the images analyzed by medical experts is unfortunately too resource-intensive and beyond the scope of this work.

Another solution common in practice is using only lossless compression but such methods never achieve anywhere near as high compression ratios (by order of magnitude) as lossy methods – for example the study [9] finds that on medical data the traditional lossless codecs hardly achieve compression ratios over 4:1, while on the test set the proposed method has average ratio over 40:1 at $\mathrm{PSNR} > 55\mathrm{dB}$. Proper research into lossy methods is therefore surely justified.

The traditional approach to image compression are hand-designed codecs that are implemented as hard-coded algorithms, based on human experience and intuition (see Sec. 2). As with many problems in image processing and computer vision in the last decade, avenues are being explored on how to learn optimal codecs from data. Modern research in learned image compression started with the works of Toderici et al. [10] as the first fully learned method applicable to large images and outperforming some established traditional codecs. A surge of interest in learned image compression came after the seminal works of Ballé et al. [11, 12] and Minnen et al. [13]. These works laid the groundwork for further research and it can be argued that most state-of-the-art (SOTA) methods nowadays are extensions of these methods.

The core structure of a learned method typically consists of an autoencoder which transforms the input and produces a latent representation of the image which will constitute the bitstream. This representation is then quantized so that it can be passed to an entropy coder which losslessly converts the discrete representation to an actual bitstream. The third integral component is an entropy model of the latent, i.e. a probability distribution model of the symbols (after quantization) of the latent representation, as this is required by the entropy coder. This pipeline can be trained end-to-end in an unsupervised manner and the minimized loss is the sum of two terms: The distortion in the image reconstruction and the entropy (i.e. expected bitrate) of the latent. The entropy coder is used off-the-shelf and is not subject to training. One of the great advantages of learned image compression is that the training is relatively simple and cheap which makes it possible to adapt a method for a particular modality, such as medical images, whereas for conventional hand-designed codecs such adaptation is not feasible.

The proposed method extends [13] to volumetric medical data consisting of individual slices, i.e. a sequence of 2D images. This type of data is acquired for example by a CT scan (see Fig. 1 for an example) or in an MRI. The individual slices are encoded in order. The transform from image data to the latent representation is done for each slice independently, but in the entropy estimation step the probability model of each slice (except the first) is conditioned on the previous slice, which enables a more accurate estimation of latent distribution since neighboring slices typically have high mutual information. This allows for higher compression ratios with no loss in the reconstruction quality. On the decoding side, the images are decoded in the same order, so that the previous slice is again available when decoding the next. Note that the proposed method works with already digitized uncompressed images in normalized intensity range (typically 8bit-16bit), it doesn't in any way enter the process of image generation by the above mentioned imaging techniques.

We show in the experimental section that this relatively simple addition outperforms considerably the baseline approach in which all slices are processed completely independently by a single image compression method. Additionally, compared to processing the full volume at once our approach requires a fraction of time and memory (in practice, it would be necessary to split the volume into small chunks and compress those separately anyway). We tested the method on a dataset consisting of CT scans of various human body parts and the proposed approach is competitive even compared to established standards such as JPEG, BPG, AVIF, and even VVC-intra.

2. Related work

For a long time, lossy image and video compression was a problem solved exclusively in the traditional way by hand-designed methods. Some of these methods, such as for example H.264 [14] or H.265 [15] video codecs or JPEG image compression [2], are now in widespread use in many areas of industry, research, or everyday life. Relatively recently, the first learned codecs appeared that were able to challenge some of the traditional methods. Arguably the biggest rise of interest started after the

works of Ballé et al. [11, 12] and later Minnen et al. [13], which laid the foundation for learned image compression. These works formulated the main rate-distortion objective in a learnable way, presented a model containing the three fundamental components now present in the vast majority of learned codecs – the autoencoder for image transform, and the hyper-prior and the context module for entropy estimation – and provided the solution for dealing with the discrete quantization in training. Subsequent methods increased the performance for example by richer/larger model architecture (e.g. using attention-like modules) [16], improved context modules [17, 18, 19], richer entropy model (e.g. Gaussian mixtures) [16], or different simulation of quantization [20, 21].

Recently, a promising research direction is coercing the reconstruction to better satisfy the expectations of the human visual system even at the expense of objective (e.g. PSNR) quality. This can be achieved for example by augmenting the loss by a term that better models human perception (such as LPIPS [22]) [19], or by training the decoder in an adversarial manner as in GANs [23, 24]. Such approaches can achieve significant bitrate savings but unfortunately are not suitable for medical data, where the reconstructed image must be objectively undistorted and not just look natural.

Literature on learned compression for medical images is relatively scarce, this area is still dominated by more traditional approaches such as compression in the wavelet domain [25]. Probably the closest match for the proposed method is the lossless compression of 3D volumes by Chen et al. [26]. In our work, however, we focus on lossy compression. Other works propose partitioning the image into relevant (for the diagnosis) and less relevant regions and apply different compression ratios there [27]. Learned lossy compression for 2D medical images is investigated for example in [28].

3. Method

The proposed approach is based on the single image compression method by Minnen et al. [13], which we extend for multi-slice volumetric images. The method [13] consists of three main components:

- An encoder/decoder which performs the transform between the input image space and the latent representation (commonly called "latent").
- A hyper-encoder/decoder (called hyper-prior) which analyzes the latent and stores a small piece of side information into the bitstream that is used later to estimate the parameters of the probability distribution of the latent (the entropy model).
- A context module that processes the image latent in an autoregressive fashion (i.e. causally) and is also a part of the entropy model parameter estimation.

The encoding and decoding branches of the pipeline are connected only via the bitstream which stores the latent and hyper-latent representation of the image. To this end the latents must be quantized, for which scalar integer rounding is used, because the entropy coder that converts the values into their corresponding bit codes can only operate on discrete data (continues values cannot be stored in the bitstream).

The advantage of the context module is that the entropy parameters can be very accurate and image-specific, the disadvantage is that the autoregressive processing does not play well with the parallel processing common in deep learning. For each new pixel to be decoded the entropy parameters must first be estimated, the pixel decoded and only then can the decoding move to the next pixel. As a result, a usually parallelized operation such as convolution cannot be computed for the whole image at once but pixel by pixel in alternation with the entropy coder. Another disadvantage is that the context prevents using the so-called mean-subtracted quantization, which will be specified in the next section. We get rid of these drawbacks in the proposed method by replacing the autoregressive context from [13] with an analogous module that runs on the previous slice in the sequence.

Model details The input to our method is a sequence of 2D slices $x^0, \dots, x^{N-1}$ (superscripts denote slices, subscripts pixel indices) which are processed in order. The transforms to and from the latent representation denoted y^i are done for each slice independently but the entropy model, i.e. the probability distribution $p_{\hat{y}}(\hat{y}^i)$ of the quantized latent $\hat{y}^i$ (hat denotes quantization operation), is conditioned on the latent of the previous slice $\hat{y}^{i-1}$. This helps decrease the entropy of $\hat{y}^i$ and therefore the necessary bitrate while avoiding the disadvantages of an autoregressive context model. It is done as follows: Instead of running the context model on the currently encoded slice in an autoregressive fashion, we run it on the (quantized) latent $\hat{y}^{i-1}$ of the previous slice. During decoding, the slices are processed in the same order so $\hat{y}^{i-1}$ has already been decoded in full and is available when $\hat{y}^i$ is being decoded and the entropy model can again use information from the previous slice. This approach does not require autoregressive processing but can instead be done in parallel for the whole slice without waiting for each new pixel to be decoded. In other words, the context module is autoregressive in the slice sequence but that does not restrict any 2D operations contained within one slice such as convolutions – instead of decoding individual pixels we can decode whole slices in parallel.

We model the distribution $p_{\hat{y}}(\hat{y}^i)$ of the quantized latent $\hat{y}^i$ by a per-dimension j (i.e. spatial pixel and channel) independent Laplace distribution with mean and scale parameters (μ^i_j, σ^i_j). These two parameters

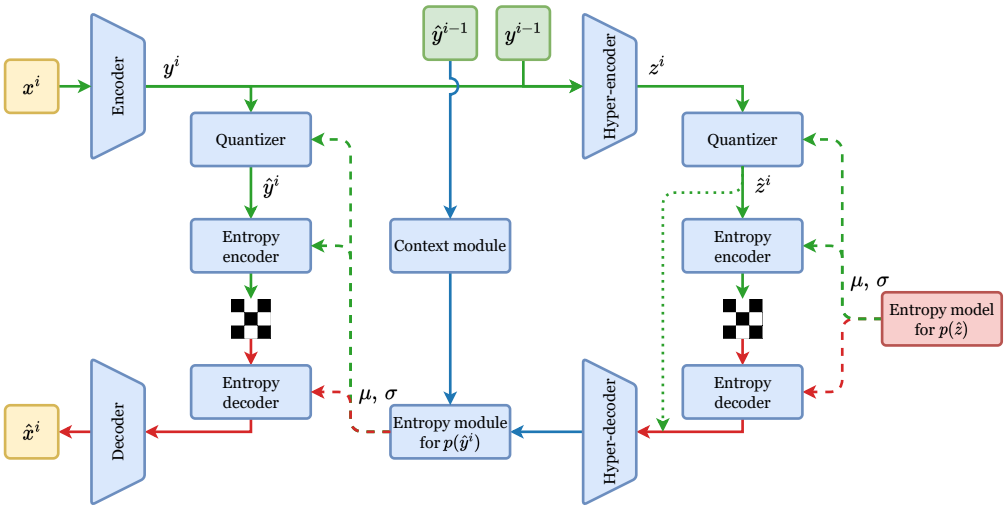

Figure 2: Overview of the proposed compression pipeline. **Connectors:** Green are operations performed only in encode, red are operations performed only in decode and blue are operations performed both in encode and decode. Checkboard denotes the bitstream. **Procedure:** The input image is passed through an encoder, producing the latent y^i. The latent is concatenated with the latent of the previous slice, y^{i-1}, and passed through the hyper-encoder, producing the hyper-latent z^i. This hyper-latent is quantized to $\hat{z}_i$ and stored using fixed entropy model $p(\hat{z})$. Parameters of the image-adaptive entropy model $p(\hat{y}^i)$ are estimated by a context that processes the previous slice's latent $\hat{y}^{i-1}$, and hyper-decoder that processes the hyper-latent $\hat{z}^i$. These two are concatenated and passed through an entropy module to produce the entropy parameters (μ^i, σ^i). The latent $\hat{y}^i$ is stored in the bitstream. In decode the hyper-decoder, context, and entropy module have to run again because the parameters (μ^i, σ^i) are required for decoding of $\hat{y}^i$ from the bitstream; for this the latent of the previous slice $\hat{y}^{i-1}$ is already available. The decoded latent $\hat{y}^i$ is passed through the decoder to produce the reconstructed image $\hat{x}^i$.

are estimated adaptively for each image i and each pixel j (incl. channels) of the latent by the hyper-prior and the context module. For quantization of the latent we use integer rounding with mean-subtraction, meaning that the value is first offset by the estimated mean of its distribution before being rounded, (image index omitted)

$$\hat{y}_j = \lfloor y_j - \mu_j \rceil + \mu_j, \tag{1}$$

where $\lfloor \cdot \rceil$ is integer rounding. This improves performance because then quantization doesn't change the mean of the distribution, but it requires that the entropy parameters of the latent are estimated before the latent is quantized. In particular, both of the entropy estimation modules (hyper-prior and context) must operate on non-quantized values y^i, otherwise an implicit relation would arise. This is difficult to achieve in a single-image autoregressive context model and for example the quantization in [13] does not use mean-subtraction, but since in the proposed method the context module uses the previous slice, using mean-subtraction is possible.

The full procedure of processing a slice x^i is illustrated in Fig. 2. The image is passed through an encoder E, producing the latent $y^i = E(x^i)$. The latent is concatenated with the latent of the previous slice, y^{i-1}, and passed through the hyper-encoder E_h, producing the hyper-latent $z^i = E_h([y^{i-1}, y^i])$. This hyper-latent is quantized, $\hat{z}_i = Q(z_i)$, so that it can be stored in the bitstream. The parameters of the entropy model of the quantized latent $\hat{y}^i$ are estimated as follows. A context module C processes the previous slice's latent $\hat{y}^{i-1}$ and hyper-decoder D_h processes the hyper-latent $\hat{z}^i$. These two are concatenated and passed through an entropy module E_p to produce the final entropy parameters $(\mu^i_j, \sigma^i_j) = E_p([C(\hat{y}^{i-1}), D_h(\hat{z}^i)])^i_j$ for each pixel j of the latent. With these parameters available the latent can be quantized and stored in the bitstream and the encoding proceeds to the next slice.

During decoding, operations responsible for estimating the entropy model $p_{\hat{y}}(\hat{y}^i)$ have to be executed again because the entropy model is required by the coder to decode $\hat{y}^i$ from the bitstream. The hyper-latent $\hat{z}^i$ is decoded first and since the latent of the previous slice $\hat{y}^{i-1}$ is already decoded and available, the estimation of the entropy parameters (μ, σ) proceeds as during encoding. Having those, $\hat{y}^i$ can be decoded and passed through the decoder D to finally produce the reconstructed image $\hat{x}^i = D(\hat{y}^i)$. The decoding then proceeds to the next slice.

What remains to specify is the entropy model $p_{\hat{z}}(\hat{z})$ of the hyper-latent $\hat{z}$, since that is also processed by the

entropy coder and stored in the bitstream. We model it by per-channel Laplace distribution, meaning that each channel of z^i has its own mean and scale parameters (μ, σ) but those are spatially constant so that the model is not tied to a fixed image resolution. These parameters are subject to training but fixed once the model has been trained (i.e. unlike $p_{\hat{y}}(\hat{y})$ it is not image-adaptive). For quantization of z we again use mean-subtracted rounding in a similar fashion as in Eq. (1).

Details of the model architecture are concisely summarized in Tab. 1.

Training details In training we optimize the rate-distortion loss L (image indices omitted)

$$L = \mathbb{E}_{x\sim p_x}\left[-\log_2 p_{\hat{y}}(\hat{y})\right] + \mathbb{E}_{x\sim p_x}\left[-\log_2 p_{\hat{z}}(\hat{z})\right] + \lambda \cdot 255^2 \cdot \mathbb{E}_{x\sim p_x}\left[\|x-\hat{x}\|_2^2\right], \tag{2}$$

where λ controls the rate-distortion tradeoff (determines approximate target bitrate) and $p(x)$, the distribution of uncompressed images, is evaluated by batch averaging. The first two terms on the right-hand side are approximate (theoretical) bitrates required by the entropy coder to encode the latents. These are used in training as an estimate of the actual bitrates because the non-differentiable entropy coders are removed from training.

Our description of $p_{\hat{y}}(\hat{y})$ and $p_{\hat{z}}(\hat{z})$ so far was somewhat simplified. The Laplace parametric density is used only as a model to conveniently parametrize the discrete distribution over the symbols after quantization. In the actual evaluation, however, we have to account for the whole interval corresponding to each discrete value because of quantization. This is done by integrating the parametric density over the corresponding interval, for example

$$p_{\hat{y}}(\hat{y}^i_j) = \int_{\hat{y}^i_j-\frac{1}{2}}^{\hat{y}^i_j+\frac{1}{2}} P_{\hat{y}^i_j}(t)dt, \tag{3}$$

where $P_{\hat{y}^i_j}$ is the continuous Laplace density model parametrized by (μ, σ) corresponding to $p_{\hat{y}}(\hat{y}^i_j)$, the discrete distribution of $\hat{y}^i_j$. In practice, this is done by using the cumulative distribution function of the Laplace density.

In each training iteration we randomly sample a small subset of n consecutive slices from each image in the batch and process those through the model as a small volume. For the first slice x^0 of this subset we calculate the latent $y^0 = E(x^0)$ using an auxiliary single-image model which shares the same encoder with the multi-slice model. For $x^1, \ldots, x^{n-1}$ we proceed as described above and these slices are used to evaluate the loss in Eq. (2). The first slice x^0 is excluded from optimization of the multi-slice model but is used to train the auxiliary single-slice model used for compression of the first slice in each volumetric series. This model has the same encoder/decoder as the multi-slice model and the same architecture (not weights) of the hyper-prior but does not include the context and entropy module – the hyper-decoder directly predicts the (μ, σ) parameters of the latent entropy model. In validation and testing, we use this auxiliary model to compress the first slice of the volume and then proceed sequentially with the multi-slice model.

Table 1
Model architecture details. *conv* is a `Conv2D` layer with kernel size k, stride s and output channels c. *transpose* is a similarly specified `ConvTranspose2D`. *GDN* and *IGDN* are the generalized divisive normalization layer [11] and its inverse, respectively. *PReLU* is the parametric ReLU [30].

Encoder: conv k5 s2 c192 → GDN → conv k5 s2 c192 → GDN → conv k5 s2 c192 → GDN → conv k5 s2 c192
Decoder: transpose k5 s2 c192 → IGDN → transpose k5 s2 c192 → IGDN → transpose k5 s2 c192 → IGDN → transpose k5 s2 c1
Hyper-encoder: conv k3 s1 c192 → PReLU → conv k5 s2 c192 → PReLU → conv k5 s2 c192
Hyper-decoder: conv k5 s2 c192 → PReLU → conv k5 s2 c288 → PReLU → conv k3 s1 c384
Context: conv k5 s1 c384
Entropy module: conv k1 s1 c768 → PReLU → conv k1 s1 c576 → PReLU → conv k1 s1 c384

The quantization operation must be approximated during training because it has zero gradient almost everywhere. For both the latents y and hyper-latents z we use the straight-through quantization [29], which performs integer rounding in the forward pass but acts as identity in the backward pass. For evaluation of the bitrate in the entropy models, however, we simulate quantization by additive uniform noise from the $(-\frac{1}{2}, \frac{1}{2})$ range. This way the hyper-decoder and decoder get the more realistic integer-rounded values (with mean-subtraction as in Eq. (1)) but the entropy estimation is calculated using the uniform noise simulation, which reportedly leads to better performance [20].

4. Results

Dataset We trained and tested the method on the Pediatric-CT-SEG dataset of CT-scan images of various organs downloaded from the Cancer Imaging Archive [6] (patient and acquisition parameters specified therein). We chose this dataset for its diverse content. The dataset consists of 359 volumetric images each with a different number of slices ranging from 41 to 1104. We randomly selected 10 of the volumetric images for testing (2184 slices in total) and the rest for training. The 2D slices

are 12bit grayscale images with a resolution of 512×512, originally stored uncompressed at 16 bits per pixel (bpp). An example slice from the dataset is in Fig. 1.

Training We trained the model on random spatial crops of size 256×256 and tested it on full-resolution images. For training, we randomly chose $n = 3$ consecutive slices as a good compromise between training speed and exploiting the sequential processing. We trained with batch size 8 using the Adam optimizer [31] with an initial learning rate of $1e-4$ for 1M iterations after which we decreased the learning rate to $1e-5$ for another 200k iterations. We trained a new model for 6 values of λ in the range from 0.032 to 3.2, which on the test set results in 0.05 to 0.65 bits per pixel, thus achieving a compression ratio of 25:1 to 320:1 with respect to the original images.

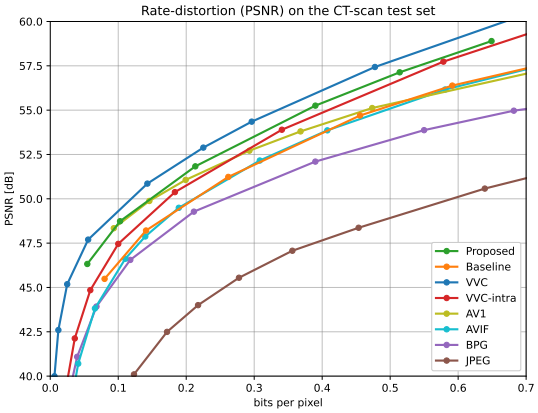

Figure 3: Rate-distortion performance of the proposed and benchmark methods on the test set of CT-scan images.

Benchmark methods We compare the performance of the proposed method with a baseline learned single-image compression model and a number of established traditional image compression methods. The single-image baseline is a learned model with the same architecture as the auxiliary model we use to compress the first slice and was trained on the same train set. Comparison with this method shows performance gain from the proposed sequential processing and the context module. The traditional methods are a broad selection ranging from well-known and established codecs commonly used in practice to the state-of-the-art prototype. Such comparison therefore well positions the proposed method in the landscape of existing methods and gives insight into its properties in potential use in practice. Below we briefly describe each of the methods used in the comparison and optionally its configuration, afterwards we provide commentary on the results summarized in Fig. 3 and Tab. 2.

Baseline is a learned single-image compression model with the same architecture as the proposed method but without the context and entropy module (the hyper-decoder directly predicts the entropy parameters). It is trained on the same train set as the proposed and uses the same training schedule. **JPEG** [2] is a well-known widely used compression method developed in the 90s. Although used for medical data and having the advantage of being very fast both in encode and decode, it is arguably not a very suitable method for such use as its performance is relatively low by today's standards. We use the implementation in `pillow`. **BPG** [3] is essentially a single-image wrapper of the intra-frame compression of the HEVC (also known as H.265) video codec. Although not widespread, it is one of the top methods currently available for everyday use. We used the `jctvc` encoder via the public BPG library configured to 12bit internal bitdepth. **AVIF** [32] is a single-image format of the AV1 video codec (essentially AV1-intra), one of today's top codecs from those that are readily available e.g. in browsers. In the comparison we used the `libaom-av1` encoder via `ffmpeg` configured to 12bit internal processing, each slice in the series is compressed individually. **AV1** [32] is a video codec in terms of quality approximately on the level of or slightly outperforming HEVC but unlike HEVC its use is royalty-free, it is therefore arguably the best video codec readily available today (with production-level encoders and decoders available). In our comparison, we used `ffmpeg/libaom-av1` in 12bit mode and compressed each volumetric image as a video sequence consisting of the individual slices. **VVC** [33] (H.266) is the best existing video codec nowadays but its development is still ongoing and the available encoders/decoders are on the prototype level and for most practical use cases prohibitively slow. Its adoption in practice, medical or otherwise, is also hindered by the fact that its use is not royalty free. We used the VTM 18 reference implementation in 12bit mode and again compressed each volumetric image as a video sequence consisting of the individual slices. **VVC-intra** [33] is the intra mode of VVC. For single-image compression, it is the best available codec nowadays but currently inherits the disadvantages listed above for VVC. We used it in the same configuration as VVC video but compressed each slice individually.

Results The rate-distortion curves of the benchmarked methods on the CT-scan test set are in shown in Fig. 3, their ranking and quantitative comparison with respect to VVC-intra is in Tab. 2 and finally, Tab. 3 shows approximate relative runtimes required to process the test set. In the testing we focused on high-PSNR range since we envision the proposed method being used primarily in the medical domain, where sliced volumetric images

Table 2
Relative bitrate increase (BD-Rate [34], negative means savings) and quality gain (BD-PSNR [34], positive means improvement) of the benchmarked methods compared to VVC-intra in the range $\mathrm{PSNR} > 45$dB.

Method	BD-Rate [%]	BD-PSNR [dB]
JPEG	+248.2	-7.57
BPG	+51.6	-2.40
AVIF	+22.7	-1.26
Baseline	+20.4	-1.14
AV1	+6.2	-0.40
VVC-intra	0.0	0.00
Proposed	-11.4	+0.64
VVC	-23.6	+1.44

Table 3
Approximate relative time required to encode and decode the full test set at $\mathrm{bpp} = .3$ compared to the proposed method ($t = 35$ seconds). Times include file I/O where unavoidable.

Method	Device	Time [t]
JPEG	CPU	2e-1
Baseline	GPU	7e-1
Proposed	GPU	1.
BPG	CPU	7e1
AVIF	CPU	1.4e2
AV1	CPU	1e3
VVC-intra	CPU	1.5e3
VVC	CPU	5.5e3

are common. Let us provide some commentary on the results.

The baseline learned method performs on the level of AVIF – the curves almost overlap. Although AVIF is undoubtedly a better codec in a general setting, the learned baseline exploits the advantage of domain specificity – it has been trained on similar CT data. BPG generally performs well on natural images where the target PSNR is usually lower, but to achieve imperceptible distortion in medical data we observed that the reconstruction PSNR should be above 55dB (for typical images with sufficient structure). We suspect there is some issue with the configuration of the encoder at high bitdepth processing because BPG obviously struggles with achieving high PSNRs. It is no surprise that JPEG cannot compete with the latest methods. VVC-intra does very well and outperforms AVIF by a large margin in the whole range. With AV1 we experienced similar problems as with BPG – it apparently "saturates" at higher bitrates and struggles to achieve high PSNR, which is possibly again some issue with the high bitdepth configuration of the encoder (although we used the same encoder as for AVIF and in that case it worked fine). But from comparison with AVIF in low to mid bitrates we can see that the sequential "video" processing of the image volume is clearly beneficial with noticeable performance gain. This conclusion is further strengthened by the results of the VVC (video) codec, which on performance alone is a clear winner of the whole comparison, outperforming all other methods (including the proposed) by a margin in the whole range.

The proposed method is significantly better than the baseline (compare green and orange curves in Fig. 3), on average achieving almost 30% rate savings (for the same quality) and 1.8dB quality increase (for the same rate). It also outperforms all image codecs such as AVIF, BPG, and especially VVC-intra, which is no small feat. This is only due to the proposed sequential context because the baseline alone is significantly below VVC-intra. It is however still a relatively small and simple model and therefore no match for VVC but we will see that in that comparison it wins on runtime.

A clear and quantitative ranking of the methods is provided in Tab. 2, which shows the average bitrate increase/savings and PSNR quality loss/gain evaluated by the BD-Rate and BD-PSNR [34], respectively. We positioned VVC-intra as the reference SOTA image codec and compared all others to it, as they perform on the test set. The table shows average bitrate savings and performance gain in the middle and right column, respectively. Only the proposed method and VVC video achieve improvement (BD-Rate is negative and BD-PSNR is positive).

Finally, in Tab. 3 we show the relative runtimes required for processing (encode and decode) the whole test set (10 volumetric images consisting of 2184 slices) with respect to the proposed method (i.e. value < 1 means the method is faster than ours, > 1 means it is slower). These runtimes are listed for $\mathrm{bpp} = .3$, approximately the middle of the tested range, since the traditional methods are slower at higher bitrates (the proposed has constant speed across the range). Here the ranking is quite different than in the performance. JPEG and of course the baseline are the only methods faster than the proposed, all others are slower and some of them quite significantly, especially the well-performing VVC which is clearly prohibitively slow. We argue that the video codecs are simply not fast enough for practical use. In fairness, the proposed method and the baseline run on GPU (though still each slice sequentially) while the traditional methods are CPU-only without any external parallelization. On the other hand, our implementation is intended only as a proof of concept and we didn't invest much effort into runtime optimization. For example, in both encode and decode the encoder and decoder process each slice independently. In testing we really process them sequentially for simplicity while it is possible to "batch" them and process in parallel (as many as the GPU memory permits), which would reduce the runtime.

Contrary to usual customs, we do not provide exam-

ples and qualitative comparison of image reconstructions because due to the high reconstruction quality and similar performance of the benchmarked methods we were not able to come up with example images that demonstrate any noticeable difference – on screen all the results look identical.

5. Conclusion

We presented an extension of a single-image learned compression method to volumetric multi-slice images with an emphasis on the medical domain, where such type of images is quite common. Although the modification is relatively simple and straightforward, it provides several benefits – namely using a context module without introducing any problems with parallel processing in the decode and using mean-subtracted quantization. Both of these improve performance without compromising the runtime. This we verified in the comparison with a number of established compression methods. The comparison shows:

- Clear performance gain with respect to the baseline due to the proposed sequential context.
- Good performance in absolute numbers with respect to the established codecs.
- Very competitive runtimes (if GPUs are allowed).

The testing was carried out with emphasis on low-error reconstruction and even at PSNR=55dB (in most cases indistinguishable from the original) the proposed method achieves an average compression ratio of 40:1 with respect to the uncompressed original. We consider these results a solid proof of concept for compression of volumetric medical data.

Nevertheless, there are a number of things which can be improved or investigated further. For example, the used baseline model is far from SOTA so higher absolute performance can be gained by adopting one of the SOTA single-image learned methods as a backbone and extending that with the proposed context model. In this work, however, we focused more on investigating the relative gains from the sequential context rather than absolute performance. Next, in decode our method currently does not permit random access (as in "show me slice 42"), the whole volume needs to be decoded sequentially from the beginning. But this can be remedied by introducing intra-frames compressed by the single-image auxiliary method we use for the first slice. If we use a GOP size of 8 (meaning at most 8 slices need to be decoded for any chosen slice), we can estimate that the performance drop in Tab. 2 would be approximately $-11.4\% \rightarrow -7.5\%$ in rate and $+0.64\text{dB} \rightarrow +0.42\text{dB}$ in PSNR, which is still solid improvement over VVC-intra with practically usable runtimes in both encode and decode.

But by looking at the results of VVC we see that further gains are undoubtedly possible and we hypothesize that those can be achieved for example by a stronger context module (ours is a rather simple stack of convolutions, not in any way input-adaptive) and possibly by introducing P-frames and B-frames as in video encoding. It is our hope that this work will motivate further research into such possibilities.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 965502.

References

[1] Radiation risk from medical imaging, https://www.health.harvard.edu/cancer/radiation-risk-from-medical-imaging, Sep 2021.

[2] G. Wallace, The JPEG still picture compression standard, IEEE Transactions on Consumer Electronics 38 (1992) xviii–xxxiv.

[3] F. Bellard, BPG Image format, https://bellard.org/bpg, 2018. Accessed: 2021-09-24.

[4] AVIF image format, https://aomediacodec.github.io/av1-avif, 2022. Accessed: 2022-12.

[5] Google, WebP Image format, https://developers.google.com/speed/webp, 2018. Accessed: 2021-09-24.

[6] Pediatric-CT-SEG, Cancer Imaging Archive, https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=89096588, Aug 2022.

[7] AOM common test conditions v2.0, https://aomedia.org/docs/CWG-B075o_AV2_CTC_v2.pdf, Aug 2021.

[8] F. Mentzer, G. Toderici, D. Minnen, S.-J. Hwang, S. Caelles, M. Lucic, E. Agustsson, VCT: A video compression transformer, 2022. URL: https://arxiv.org/abs/2206.07307.

[9] J. Kivijärvi, T. Ojala, T. Kaukoranta, A. Kuba, L. Nyúl, O. Nevalainen, A comparison of lossless compression methods for medical images, Computerized Medical Imaging and Graphics 22 (1998) 323–339.

[10] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, M. Covell, Full resolution image compression with recurrent neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[11] J. Ballé, V. Laparra, E. P. Simoncelli, End-to-end optimized image compression, in: International Conference on Learning Representations, 2017.

[12] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, N. Johnston, Variational image compression with a scale hyperprior, in: International Conference on Learning Representations, 2018.

[13] D. Minnen, J. Ballé, G. D. Toderici, Joint autoregressive and hierarchical priors for learned image compression, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018.

[14] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the h. 264/avc video coding standard, IEEE Transactions on circuits and systems for video technology 13 (2003) 560–576.

[15] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the High Efficiency Video Coding (HEVC) standard, IEEE Transactions on Circuits and Systems for Video Technology 22 (2012) 1649–1668.

[16] Z. Cheng, H. Sun, M. Takeuchi, J. Katto, Learned image compression with discretized gaussian mixture likelihoods and attention modules, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[17] D. Minnen, S. Singh, Channel-wise autoregressive entropy models for learned image compression, in: 2020 IEEE International Conference on Image Processing (ICIP), 2020, pp. 3339–3343.

[18] D. He, Y. Zheng, B. Sun, Y. Wang, H. Qin, Checkerboard context model for efficient learned image compression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 14771–14780.

[19] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, Y. Wang, Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 5718–5727.

[20] L. Theis, W. Shi, A. Cunningham, F. Huszár, Lossy image compression with compressive autoencoders, in: International Conference on Learning Representations, 2017.

[21] Z. Guo, Z. Zhang, R. Feng, Z. Chen, Soft then hard: Rethinking the quantization in neural image compression, in: International Conference on Machine Learning, PMLR, 2021, pp. 3920–3929.

[22] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[23] F. Mentzer, G. D. Toderici, M. Tschannen, E. Agustsson, High-fidelity generative image compression, Advances in Neural Information Processing Systems 33 (2020).

[24] D. He, Z. Yang, H. Yu, T. Xu, J. Luo, Y. Chen, C. Gao, X. Shi, H. Qin, Y. Wang, Po-elic: Perception-oriented efficient learned image coding, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2022, pp. 1764–1769.

[25] T. Bruylants, A. Munteanu, P. Schelkens, Wavelet based volumetric medical image compression, Signal Processing: Image Communication 31 (2015) 112–133.

[26] Z. Chen, S. Gu, G. Lu, D. Xu, Exploiting intra-slice and inter-slice redundancy for learning-based lossless volumetric image compression, IEEE Transactions on Image Processing 31 (2022) 1697–1707.

[27] M. U. A. Ayoobkhan, E. Chikkannan, K. Ramakrishnan, Feed-forward neural network-based predictive image coding for medical image compression, Arabian Journal for Science and Engineering 43 (2018) 4239–4247.

[28] D. Mishra, S. K. Singh, R. K. Singh, Lossy medical image compression using residual learning-based dual autoencoder model, in: 2020 IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), 2020, pp. 1–5.

[29] Y. Bengio, Estimating or propagating gradients through stochastic neurons, 2013. URL: https://arxiv.org/abs/1305.2982.

[30] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.

[31] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR (Poster), 2015.

[32] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, et al., A technical overview of AV1, Proceedings of the IEEE 109 (2021) 1435–1462.

[33] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, J.-R. Ohm, Overview of the Versatile Video Coding (VVC) standard and its applications, IEEE Transactions on Circuits and Systems for Video Technology (2021) 1–1.

[34] G. Bjontegaard, Calculation of average PSNR differences between RD-curves, VCEG-M33 (2001).