Machine Learning with Applications 8 (2022) 100273

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

# Sampling strategies for learning-based 3D medical image compression

Omniah H. Nagoor [a,*], Joss Whittle [b], Jingjing Deng [a], Benjamin Mora [a], Mark W. Jones [a]

[a] Swansea University, The Department of Computer Science, Swansea, SA1 8EN, UK
[b] Rosalind Franklin Institute, Oxford, OX11 0FA, UK

ARTICLE INFO

*Keywords:*
3D predictors
Deep learning
Lossless compression
Medical image compression
Sequence prediction model
LSTM

ABSTRACT

Recent achievements of sequence prediction models in numerous domains, including compression, provide great potential for novel learning-based codecs. In such models, the input sequence's shape and size play a crucial role in learning the mapping function of the data distribution to the target output. This work examines numerous input configurations and sampling schemes for a many-to-one sequence prediction model, specifically for compressing 3D medical images (16-bit depth) losslessly. The main objective is to determine the optimal practice for enabling the proposed Long Short-Term Memory (LSTM) model to achieve high compression ratio and fast encoding–decoding performance.

Our LSTM models are trained with 4-fold cross-validation on 12 high-resolution CT dataset while measuring model's compression ratios and execution time. Several configurations of sequences have been evaluated, and our results demonstrate that pyramid-shaped sampling represents the best trade-off between performance and compression ratio (up to 3×). We solve a problem of non-deterministic environments that allow our models to run in parallel without much compression performance drop.

Experimental evaluation was carried out on datasets acquired by different hospitals, representing different body segments, and distinct scanning modalities (CT and MRI). Our new methodology allows straightforward parallelisation that speeds-up the decoder by up to 37× compared to previous methods. Overall, the trained models demonstrate efficiency and generalisability for compressing 3D medical images losslessly while still outperforming well-known lossless methods by approximately 17% and 12%. To the best of our knowledge, this is the first study that focuses on voxel-wise predictions of volumetric medical imaging for lossless compression.

## 1. Introduction

Millions of medical scans are produced in the UK alone each year from various modalities, including Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) (Dixon, 2019). High-resolution images are vital for patient-evaluation and clinical applications, including preoperative planning (Anagnostakos, et al., 2019), but can present a technical challenge from large storage requirements (e.g. a 3D volumetric CT or MRI image can use a Gigabyte). Besides, these massive volumes influence not only available storage space but also raise the difficulty of data streaming and communication. This problem can be addressed by employing efficient compression methods. Due to data reliability and accuracy, lossy reduction methods are generally not advised for medical image applications since it can affect diagnostic performance (Koff & Shulman, 2006; Patidar, Kumar, & Kumar, 2020). We therefore focus on lossless compression and deep learning, which achieves remarkable gains over the classical state-of-the-art for both lossy and lossless compression methods. For predictive compression schemes, various local sampling grids can be applied. For deep learning predictor-based models, the input sequence shape and size play a crucial role in learning a mapping function from the input data of known causal neighbours to the target output of the next unknown value. Generally, there is a trade-off between the sequence size and the computational cost of a model. For 3D data, various block coverage around the target voxel can be applied (Nagoor, Whittle, Deng, Mora, & Jones, 2020a, 2020b). This work examines further options for the causal neighbouring sequence and determines the compression trade-offs between size and speed. LSTMs are widely employed for solving sequence and time series prediction problems due to their effectiveness and training stability in addressing vanishing gradient problems compared to other recurrent unit alternatives. LSTM was chosen as it practically balances architecture compactness while capturing long-term dependencies and correlations within spatial neighbourhood voxel sequences. The primary intention is to comprehensively investigate

* Corresponding author.
*E-mail addresses:* 885020@swansea.ac.uk (O.H. Nagoor), joss.whittle@rfi.ac.uk (J. Whittle), j.deng@swansea.ac.uk (J. Deng), b.mora@swansea.ac.uk (B. Mora), m.w.jones@swansea.ac.uk (M.W. Jones).
*URLs:* https://www.swansea.ac.uk/staff/science/compsci/deng-j/ (J. Deng), https://www.swansea.ac.uk/staff/science/compsci/mora-b/ (B. Mora), https://www.swansea.ac.uk/staff/science/compsci/jones-m-w/ (M.W. Jones).

https://doi.org/10.1016/j.mlwa.2022.100273
Received 3 November 2021; Received in revised form 13 January 2022; Accepted 5 February 2022
Available online 11 February 2022

the best sequence pattern that would lead to optimal compression quality and performance. Therefore, all the remaining experimental settings were fixed to keep the comparisons and benchmarks consistent and equitable, including the model selection, model architecture, and training hyperparameters. To the best of our knowledge, this is the first extensive research that focuses on the voxel-wise prediction for a high resolution volumetric medical imaging lossless compression.

This paper investigates various strategies for generating neighbourhood sequences to find the best pattern, which would lead to optimal compression quality and performance for volumetric medical scans. Briefly, the main contributions of this work are as follows:

1. We present recurrent neural network (LSTM) learning-based models for compressing 16 bit medical data and offer parallelisation of the decoder resulting in a speed-up of up to 37× compared to previous methods.
2. We demonstrate a comprehensive study on sampling strategies and how they influence compression performance (compression time and compression ratio). The strategies include competing neighbourhood sampling sequences and the extraction of training sample batches from the 3D scans.
3. We demonstrate a comprehensive comparison study of the various strategies, including against state-of-the-art lossless compression methods. We outperform other methods for compression ratio and speed.

The remainder of this paper is organised as follows: Section 2 outlines the current state-of-the-art approaches for lossless compression and deep learning predictive models. Section 3, describes the proposed methodology and strategies. The experimental results are introduced and discussed in Section 4 with compression results of our proposed models compared to the state-of-the-art lossless compression methods. Finally, Section 5 concludes the main findings of this paper.

## 2. Related work

Compression is commonly described as a reduction in the bit rate required to represent data. Entropy is a unit used to measure the minimum number of bits required on average to represent a symbol belonging to a stream according to Shannon (Bishop, 2007; Goodfellow, Bengio, & Courville, 2016; Shannon, 1948). The level of compression that can be achieved depends on the type of patterns and assumption that can be made about the target data such as spatial, coding and spectral (psycho-visual) redundancy (Sridhar, 2014). According to current compression research trends, reduction methodologies can be classified into four main types: prediction-based (Lucas, Rodrigues, da Silva Cruz, & de Faria, 2017; Magli, Olmo, & Quacchio, 2004; Matsuda, Mori, & Itoh, 2000; Nagoor et al., 2020a, 2020b; Rhee, Jang, Kim, & Cho, 2020; Schiopu, Huang, & Munteanu, 2020; Schiopu & Munteanu, 2020a, 2020b; Sullivan, Ohm, Han, & Wiegand, 2012; Weinberger, Seroussi, & Sapiro, 2000; Wu & Memon, 1996, 2000), transform-based (Schelkens, Munteanu, Tzannes, & Brislawn, 2006; Taubman & Marcellin, 2002), quantisation-based (Kingma, Abbeel, & Ho, 2019; Townsend, Bird, & Barber, 2019), and end-to-end compression frameworks (Hu, Yang, Ma, & Liu, 2020). This paper will highlight the current research trends of the prediction-based paradigm, including classical (non-learned) lossless compression literature and deep learning compression literature.

Prediction-based compression is applied to reduce redundancy, wherein a decorrelation of the causal neighbouring values is applied to predict a target value. When compression is applied over a 2D varying signal, the method is known as image-codec. 3D data can be considered either a stack of 2D frames over time (video-codec) or a 3D varying signal (volume-codec). After iteratively applying a compression model to create a map of predicted values, a residual error is usually computed to measure the difference between predictions and the ground truth values. This prediction error will be further compressed losslessly using an entropy coder to reduce the coding redundancy, such as arithmetic coding, context adaptive binary arithmetic coding, Asymmetric Numeral Systems (ANS), or Huffman coding. Various linear or non-linear combinations of causal neighbourhood values along with the number of neighbours and the shape of their sampling patterns are applied within the predictive-based lossless compression literature, spanning both classical and learning-based predictive approaches.

### 2.1. Classical prediction based methods

Among the classical predictor-based methods, and within the image-codec category, the Joint Photographic Experts Group-Lossless (JPEG-LS) utilises the immediate three pixels neighbourhood to predict a target pixel applying a mode-selection scheme with the LOCO-I algorithm (Weinberger et al., 2000). A more complicated technique for image compression, which employs six pixels of the causal neighbouring into context-based Gradient Adjusted Predictor (GAP) known as Context Based Adaptive Lossless Image Codec (CALIC) (Wu & Memon, 1996). Minimum-Rate Predictor (MRP) has a wider causal neighbourhood and is an adaptive predictive-based method that applies 2D-block classification (Matsuda et al., 2000). To compress higher dimensional volumetric data, including videos and 3D medical images, several classical image coders extended their functionality to 3D space. Both (3D-CALIC) (Wu & Memon, 2000) and (M-CALIC) (Magli et al., 2004) are extended versions of the image coder CALIC. 3D-CALIC is an enhanced version supporting context decorrelation for both inter-band and intra-band modelling. On the other hand, the M-CALIC algorithm outperforms 3D–CALIC in decorrelating hyperspectral data with multiband lossless and near-lossless compression. 3D-MRP similarly extended the MRP algorithm to utilise 3D causal neighbourhood pixels and provides an enhanced error estimation, and context estimator for both 8 and 16 bit depth contents (Lucas et al., 2017).

A well-known 3D codec for video compression is High Efficiency Video Coding (HEVC) (Sullivan et al., 2012), which combines numerous coding tools and provides compression for both lossy and lossless options. The lossless mode is a predictive-based scheme that applies both inter and intra-prediction to reduce data redundancies within and between frames. HEVC lossless mode applications include 3D medical imagery using Range Extension (Bossen, Flynn, Sharman, & Sühring, 2019) with 4:0:0 chroma format for one channel component 16-bit data. In summary, most of the classical lossless compression approaches rely on hand-crafted or linear combinations with a few causal neighbouring pixels coverage for their predictions. Moreover, such methods are designed to perform well only on specific data domain for which they were intended, most commonly natural images or video sequences. These main limitations demand novel approaches with more flexibility in estimating non-linearity. The deep learning approaches form a great potential and promising research direction that provides both efficacy and flexibility to represent non-linear data distribution.

### 2.2. Learning based prediction methods

Compared to classical state-of-the-art compression methods, the current deep learning methods are gaining remarkable compression results for both lossy and lossless compression (Hussain, Al-Fayadh, & Radi, 2018; Liu, Li, Lin, Li, & Wu, 2020; Yi, Walia, & Babyn, 2019). This shift towards the learning-based methods is consequent to its outstanding results performance in many domains, exceptional ability in representing nonlinearity, and GPU utilisation. Numerous learning-based approaches are proposed for lossy compression for applications including image-super resolution (Lai, Huang, Ahuja, & Yang, 2017), dimensionality reduction (autoencoders) (Toderici, et al., 2017), generative compression (Santurkar, Budden, & Shavit, 2018), and end-to-end compression frameworks (Hu et al., 2020). An autoregressive model is one of the state-of-the-art models in estimating data distribution and pixels likelihood. In PixelRNN (van den Oord, Kalchbrenner, &

Kavukcuoglu, 2016), the probability of each pixel conditionally depends on the probability distributions of all the previous pixels for each channel, which results in the pixel generating process to be relatively slow due to the sequential implementation. However, the PixelCNN (van den Oord et al., 2016) provides a parallelised version, employing a smaller receptive field but not fully utilising the available context. PixelCNN++ (Salimans, Karpathy, Chen, & Kingma, 2017) and Multiscale-PixelCNN (Reed, et al., 2017) are further enhanced versions utilising both parallelisation and context employment with some regularisations.

Sequence models are a particular type of supervised learning algorithm, which employ a predictive scheme. This neural network offers outstanding flexibility for various sequential input and output of arbitrary lengths that a model can maintain. Based on the length of the processed sequences, this model can be categorised into numerous types, including one-to-one, many-to-one, one-to-many, and many-to-many sequence prediction models. The domains for such models are numerous for sequential and higher-dimensional data, including sequence prediction, sequence generation, sequence classification, and sequence-to-sequence prediction. A few examples of applications are weather forecasting, product recommendation, stock market prediction, sentiment analysis, text translation, image caption, and text generation (Brownlee, 2017). Some state-of-the-art sequence prediction models contain an internal state or memory unit, which helps to learn the long-term temporal contextual information. Generally, when solving a sequence prediction problem, a model learns a mapping function $f(x_t)$, which maps an input series $x_t$ to an output sequence $y_t$. Examples of the state-of-the-art deep learning sequence prediction models are Recurrent Neural Network (RNN) (Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010), Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), Gated Recurrent Units (GRU) (Chung, Gülçehre, Cho, & Bengio, 2014), and Transformers (Devlin, Chang, Lee, & Toutanova, 2019; Parmar, et al., 2018; Radford, et al., 2019).

Although only a few contributions have been made to address lossless compression within the deep learning literature, this area is gaining more attention recently. Schiopu and Munteanu proposed a novel hybrid lossless image codec with a predictive paradigm that utilises large causal neighbouring pixels to predict a target output (Schiopu & Munteanu, 2020a). Their method also includes a residual error block to further exploit the pixel's inter-prediction and a novel context-based bit entropy coder that outperforms the traditional state-of-the-art lossless codecs. The same authors proposed an enhanced version with a different NN architecture, resulting in better efficiency and better predictions (Schiopu & Munteanu, 2020b). Another image codec that manipulates neighbouring pixels known as a channel-wise progressive prediction was presented in Rhee et al. (2020). Their proposed network is a Multilayer Perceptron (MLP), whereas a progressive training scheme is applied on both residual and channel-wise. Additionally, an Adaptive Arithmetic Coder (AAC) is used to encode the error based on the coding context. Compared to engineered codecs, the results of this MLP-based approach outperform standard image codecs in all test datasets by a significant margin. A more recent lossless compression method that employs a CNN as a predictor for video coding was produced by Schiopu et al. (2020). This block-wise compression approach replaced all intra-prediction modes of the HEVC with deep learning CNNs and gained an average bit reduction of 5% compared to the standard HEVC. Another deep learning lossless compressors that apply predictive scheme but for sequential data are DeepZip (Goyal, Tatwawadi, Chandak, & Ochoa, 2019), LSTM-Compress (Knoll, 2020), and Dzip (Goyal, Tatwawadi, Chandak, & Ochoa, 2020). In these compression frameworks, a combination of neural network-based compressor and arithmetic coding is utilised. Both DeepZip and LSTM-Compress involve RNN-based models (e.g. GRU or LSTM models) as probability estimators along with an arithmetic coding unit and specifically employed for losslessly compressing text and Genomic datasets. DZip is a more general-purpose NN-based model for reducing various dataset types using a hybrid training approach. Compared to these methods, our proposed models have a noticeably faster encoding/decoding computation time and achieve a better bit reduction compared to Knoll (2020) (see Section 4 for more details).

Nagoor et al. (2020a, 2020b) recently propose learning-based predictive methods, which supports lossless compression for 3D medical imaging (16 bit depths). Two unique 3D shapes of the surrounding neighbouring voxels (e.g. 3D cube and 3D pyramid) were applied to train MLP and LSTM models, respectively. These two approaches differ from the aforementioned learning-based codec as they offer a voxel-wise prediction model with a 3D contextual neighbouring voxels to predict a single target voxel. This paper builds on these prior many-to-one sequence models by comprehensively studying the effect of numerous causal neighbouring voxels over different dimensions and coverage (shapes) on compression performance (bpp) and compression time. Compared to most deep learning approaches, the prior and current network model architectures are relatively small (i.e. only 810 Kilobytes (KB)), but have sufficient capacity and achieve the best bit reduction compared to other classical compression approaches.

Additionally, one of the main contributions in this work is a significant improvement of the decoder procedure compared to other lossless options in terms of compression quality and performance. Evaluation results are compared to state-of-the-art lossless compression methods over various 3D datasets.

## 3. Methodology

### 3.1. Problem description

Given a data distribution defined over $V \subset R^N$, we extract several causal neighbouring sequences $X_n$ for training, where $X_n \in V$. Each sequence $X_i$ has a set of observations of surrounding neighbouring voxels' intensities $X_i = \{x_0, x_1, x_2, \ldots, x_{l-1}\}$ with a fixed sequences' length $l$ as illustrated in Fig. 1. The LSTM model is expected to learn a differentiable mapping function $\hat{y}_i = f(X_i)$ that maps the sequence of intensity values $X_i$ to a prediction of the next single target voxel value $\hat{y}_i$. While training, the LSTM predictor model learns to minimise the difference between the prediction value $\hat{y}_i$ and the ground truth value $y_i$ through backpropagation. When evaluating the model, the residual or prediction error $E = y_i - \hat{y}_i$ is computed. This volumetric prediction error is then compressed losslessly to lower bit rate using an arithmetic coder. To recover the original volume within the receiver side, an arithmetic decoding is applied to decompress the volumetric error $E$. The LSTM model will then auto-regressively generate the prediction values $\hat{y}_i$ that are summed to the residuals $E$ (see Fig. 4 for more details).

### 3.2. Causal neighbouring sequence

In a predictive-based model, the input sequence plays an essential role in learning the mapping function to the target output. Naturally, there is a trade-off between the amount of information a sequence produces to the LSTM model and the computational cost. Commonly, the longer the length of a sequences is, the slower the model gets, and the less stable the training becomes with a higher chance of facing gradient problems. This study examines and proposes various options of surrounding neighbouring sequence extracted from 3D high-resolution medical volumes. Two different block shapes were introduced: the cube and the pyramid shapes, as illustrated in Fig. 2. For each shape, several sizes (various distances to the target voxel) were applied. Voxel sequences with 1D, 2D and 3D coverage were tested.

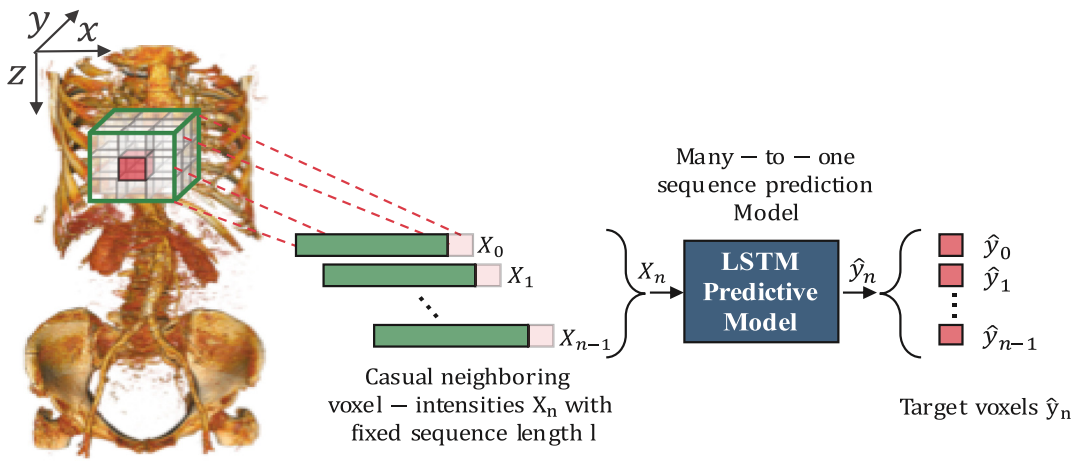



**Fig. 1.** Illustration of the supervised learning LSTM model with an explicit overview of the method for extracting the causal neighbouring sequence from 3D medical images (16 bit-depth).

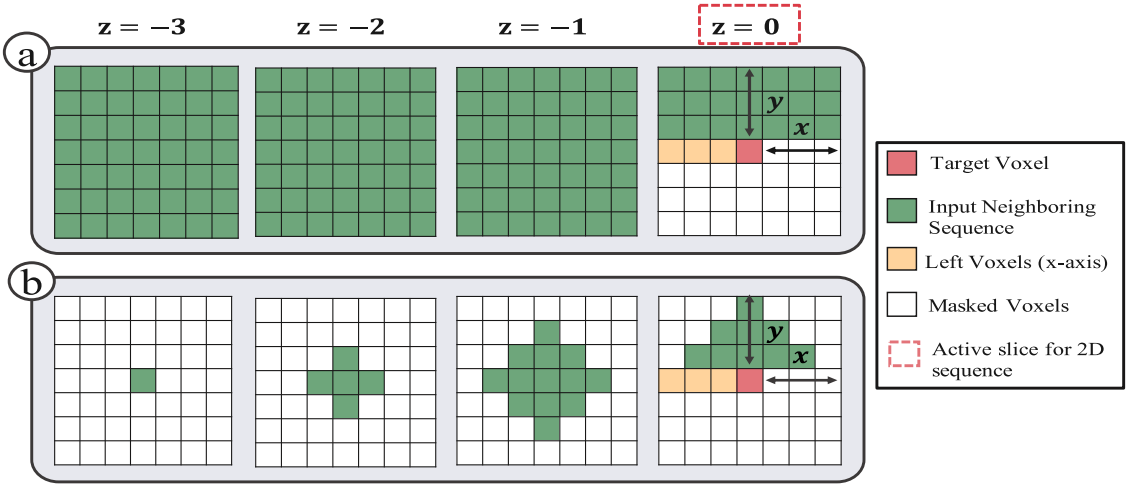



**Fig. 2.** A demonstration of the proposed causal neighbouring sequences that can have different shapes, dimensions, block sizes, and sequence lengths. The two main shapes are (*a*) Cube and (*b*) Pyramid. The red voxel refers to the target voxel to be predicted while the green voxels form the model's input sequence and the white voxels are masked (excluded). $z = 0$ is the current slice which is also the only active slice when extracting a 2D sequence. The left (yellow) pixels were included in the training for experiment 3.4.1, but omitted from the input sequences of experiment 3.4.2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

### 3.3. Model

We formulate the lossless compression problem as a supervised machine learning task. This proposed solution is also known as a many-to-one prediction model, whereas a mapping from the input sequence to the target output is learned by the LSTM model through the backpropagation process. The intention of choosing the LSTM model is that it is explicitly designed for sequence prediction problems and its ability to handle gradient problems (e.g. exploding or vanishing gradients) better than the RNN by utilising the gating mechanism. These features allow LSTM to learn long time dependency with more stable training. For a sequence prediction model, the input list forms an essential part in learning the mapping to the objective output. Fundamentally, a sequence is defined as an ordered set of observations that pass sequentially through the hidden cells of LSTM.

#### 3.3.1. Training hyper parameters

**Architecture:** The proposed model is composed of a single LSTM layer containing 128 units followed by a linear output layer used as the main architecture with the same weights for both encoder and decoder. The storage size required for the model's weights is only 276 KiloBytes (KB), while the complete model's size (weights and training hyper parameters) is 810 Kilobytes (KB). **Optimiser:** We use Adam optimiser (Kingma & Ba, 2015), with parameters $\beta_1 = 0.9, \beta_2 = 0.98$, a learning rate of $1e-4$, and a batch size of 128 for all the proposed models. **Loss Function:** a joint loss $L_{joint}$

$$L_{Joint} = MAE + \lambda(1 - |PCC|) \tag{1}$$

which is the weighted sum of the Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{2}$$

with the Pearson Correlation Coefficient (PCC)

$$PCC = \frac{cov(y_i, \hat{y}_i)}{\sigma_{y_i} \sigma_{\hat{y}_i}} \tag{3}$$

where $y_i$ is the ground truth voxel value, $\hat{y}_i$ is the model's prediction, $n$ is the total number of data samples, $cov$ is the covariance, $\sigma_{y_i}$ and $\sigma_{\hat{y}_i}$ are the standard deviation of $y_i$ and $\hat{y}_i$. Since PCC measures the statistical relationship between two continuous variables, we found that integrating it with MAE has a great impact on both accuracy and stability for solving our regression problem.

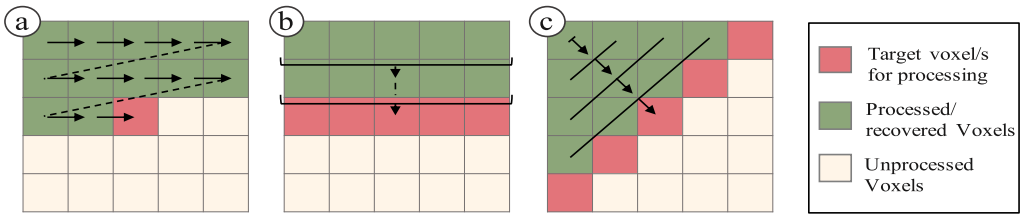



**Fig. 3.** The order at which the decoder will decode voxels based on the defined causal neighbourhood specifications. (*a*) The order of decoding voxels (sequentially) restricted by neighbourhood dependencies applied in 3.4.1. (*b*) The order of decoding voxels (in parallel) by decoding an entire batch applied in 3.4.2. (*c*) a possible parallel approach to decoding voxels, which does not allow full GPU occupancy.

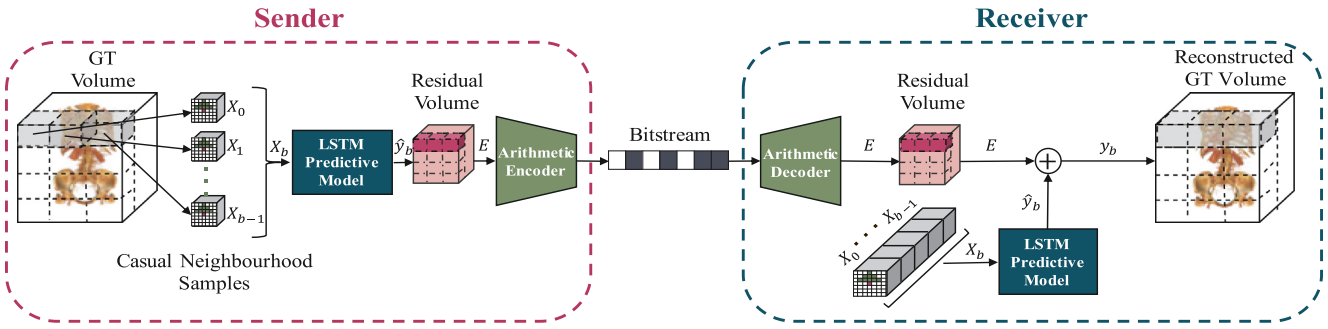



**Fig. 4.** Overview of the optimised version of our lossless compression framework using LSTM. This version leverages parallelism by employing a novel input sequence, which adds more flexibility and allows the decoder to process several batches of sequences in parallel.

### 3.4. Benchmarks

This section presents each of the benchmarks we investigate, including motivation, rationale, experimental settings, and evaluation metrics. This extensive study aims to establish the local sampling grid and sampling scheme that allow the RNN model to achieve high compression ratio and fast encoding–decoding performance for 3D medical images. All models used in this paper are standard Long Short Term Memory Networks (LSTM) with a single hidden layer with 128 units and a linear output layer. An evaluation procedure of 4-fold cross-validation on 12 high-resolution CT volumes of patients' Torso from **Dataset1** was applied. Each of the proposed models was trained using one of the sampling patterns with the same training parameters, and an equal number of training-steps (60 epochs). The results are reported using bits-per-pixel/voxel storage for each model. Also, we further tested the proposed pre-trained models on completely unseen public MRI volumes from **Dataset2**.

#### 3.4.1. Benchmark 1: Optimal input sequence (shape and size)

Sequence prediction using the described approach produces a competitive compression ratio. In this experiment, we evaluate many alternative strategies for selecting the local sampling grid to produce optimal sequence prediction. Within this benchmark, we examine 15 local sampling grid options around the target voxel, including causal voxels values from different dimensions. In the 1D case, only the previous five left voxels (voxels from only the x-axis) are used. The 2D case has four options: two with a cube shape and another two for the pyramid shape each with $(13 \times 13)$ and $(11 \times 11)$ block sizes. In the 2D cases, only local neighbourhood voxels from the same slice are included. For the 3D case, numerous options have been provided, including five samples for each shape to find the optimal input vector. In the 3D cube block, voxels with the following block sizes: $(3^3)$, $(5^3)$, $(7^3)$, $(9^3)$, and $(11^3)$ were extracted. For the pyramid shape, various distances to the target voxel were utilised including $(5 \times 5, 3 \times 3, 1)$, $(7 \times 7, 5 \times 5, 3 \times 3, 1)$, $(9 \times 9, 5 \times 5, 3 \times 3, 1)$, $(9 \times 9, 7 \times 7, 5 \times 5, 3 \times 3, 1)$, and $(13{\times}13, 9{\times}9, 5{\times}5, 3{\times}3, 1)$. A demonstration of the proposed neighbouring sequences that have different shapes (*a*) cube and (*b*) pyramid are shown in Fig. 2, where the red voxel refers to the target voxel to be predicted, the green voxels form the model input sequence, and the white voxels are masked (i.e. excluded). $z = 0$ is the current slice which is also the only active slice when extracting a 2D sequence. In the novel input sequences, the yellow voxels will be removed. Sequence values are normalised to the range $[-1, 1]$ before applying the LSTM model.

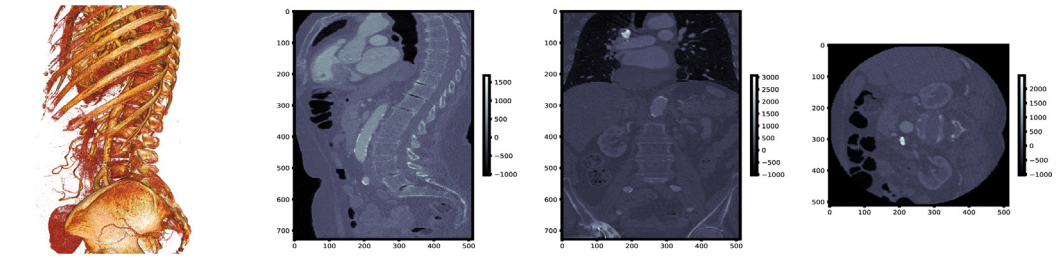
#### 3.4.2. Benchmark 2: Optimising decoder performance

Additionally, we propose a novel input sequence with 14 local sampling grid options to optimise the decoding time by leveraging parallelism. Other learning-based lossless compression approaches (Goyal et al., 2019, 2020) need to run in a deterministic environment during compression and decompression to produce the correct lossless results. Due to hardware or framework limitations, these methods perform encoding and decoding on a single CPU thread to guarantee deterministic computation. In Nagoor et al. (2020a, 2020b) only the decoder needs to run in a deterministic environment due to the sequential nature of voxels' dependencies (generating the neighbouring voxels before processing the next target) as shown in Fig. 3(a). A benefit of our new proposed approach is that we leverage parallelism and take full advantage of GPU acceleration while ensuring that both encoder and decoder are running in a deterministic fashion.

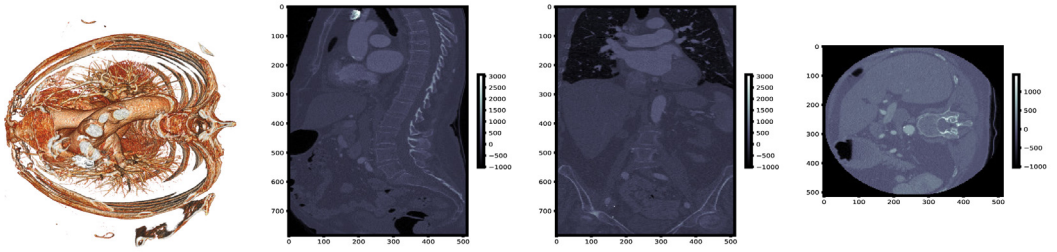
We adjust the input sequence to allow parallelism and retain favourable compression performance. We introduce a reduced version of all the neighbourhood sequences proposed in the previous Section 3.4.1. In these sequences, the left $x$-axis voxels (yellow voxels in Fig. 2) are removed from the input sequence. This allows the decoder to process batches of sequences in parallel, as illustrated in Fig. 3(b). Removing the left voxels forms a simple but effective strategy to leverage parallelism compared to decoding voxels diagonally Fig. 3(c). Parallel implementation using diagonal voxels does not lead to optimal GPU occupancy compared to our approach and complicates implementation.

#### 3.4.3. Benchmark 3: Encoding-decoding performance

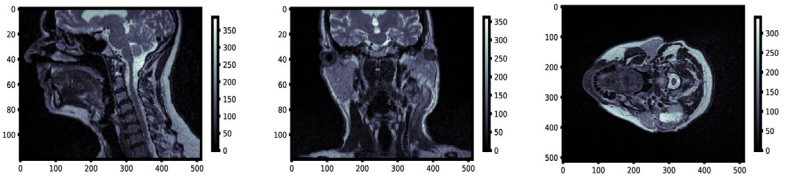
This benchmark demonstrates the computation time in seconds (s) to compress and decompress the same file with each model trained on a unique causal sequence. All experiments have been conducted on one machine with NVIDIA GeForce GTX 1080 GPU and Intel(R) Core(TM) $i7 - 4770K$ CPU.



(a) Volume 1 from **Dataset1**



(b) Volume 6 from **Dataset1**



(c) Volume 1 from **Dataset2**

**Fig. 5.** A visualisation of some (16-bits) sample volumes from the two datasets used in this paper, namely, **Dataset1** and **Dataset2** is shown. Fig. 5(a) and (b) illustrates 3D volume visualisation of patient's entire Torso with three orthogonal slice views (axial, sagittal and coronal) of two sample volumes from **Dataset1**. While Fig. 5(c) presents three orthogonal slice views (axial, sagittal and coronal) of a single sample MRI volume from **Dataset2** illustrating the patient's head and neck.

#### *3.4.4. Benchmark 4: Sampling strategy*

Approximately one billion voxels are available in the dataset, making training on all of the samples costly. This benchmark examines various sampling strategies for selecting training voxels from volumetric CT scans. Three main strategies were investigated, namely, random sampling, Gaussian sampling, and slice-based sampling. The intention is to find whether there is any preference for generating training samples as a representative subset of the available voxels in a way that benefits the compression performance. For all these sampling schemes, the same number of training samples were used, with a total of approx 4.7M unique training samples and the same causal neighbouring sequence (3D pyramid, $(13\times13, 9\times9, 5\times5, 1)$, $N = 175$) was applied as the sequence shape for each strategy.

A uniform selection across multiple volumes is applied in random sampling, where each voxel has the same probability of being selected. In contrast, Gaussian sampling performs a biasing scheme towards the centre of the volume, so voxels in mid-slices will have higher probabilities than the edge voxels. The slice-based scheme extracts multiple complete 2D slices across the volume $z$-axis with a fixed stride (fixed interval), where the aim is to fully sample individual cross-sections and massively reduce the number of samples taken.

#### *3.4.5. Benchmark 5: Compression improvement during training*

Evaluation of compression ratio in bpp for the trained models is provided in this examination. To clarify, we want to measure the improvement of compression quality during training time for five different models by compressing the same volume. The evaluation was conducted at specific epochs during training namely, 10, 20, 30, 40, 50, and 60 epochs. Each model was trained on a unique causal neighbouring sequence.

#### *3.4.6. Benchmark 6: Comparing with state-of-the-art lossless compression methods*

In this benchmark, the trained models with the best compression results from benchmarks 3.4.1 and 3.4.2 are compared to existing state-of-the-art approaches for lossless compression of volumetric medical data. Compression ratios for all of the compared approaches are reported using bits per pixel over the two available datasets.

## 4. Experimental results

### *4.1. Dataset*

The first dataset used in this study contains 12 high-resolution CT private volumes of human Torso generated by a local hospital known as **Dataset1**. Each volume in this set is stored as 16-bit greyscale Digital Imaging and Communications in Medicine (DICOM) images. All scans have $512 \times 512$ resolution, $[.488, .488]$ pixel spacing, and .625 mm slice thickness. The number of images per volume varies $z \in [728, 1008]$. The minimum intensity value is $-1024$ in all volumes, and the maximum is 3071. An illustration of orthogonal slice views and 3D volume rendering of two sample volumes from **Dataset1** are presented in Fig. 5(a) and (b).

Another dataset, which was involved only for conducting experimental evaluation, is a public dataset denoted as **Dataset2**. **Dataset2** contains DICOM files that form a total of 12 MRI volumes of patients' head and neck scans (Cardenas, et al., 2019, 2020; Clark, et al., 2013). Each image has $512 \times 512$ resolution with $[.5, .5]$ pixel spacing, and 2 mm slice thickness. The total number of frames per volume is 120 with a minimum intensity value of 0 and a maximum of 689. An illustration of orthogonal slice views of one sample volume from **Dataset2** is shown in Fig. 5(c).

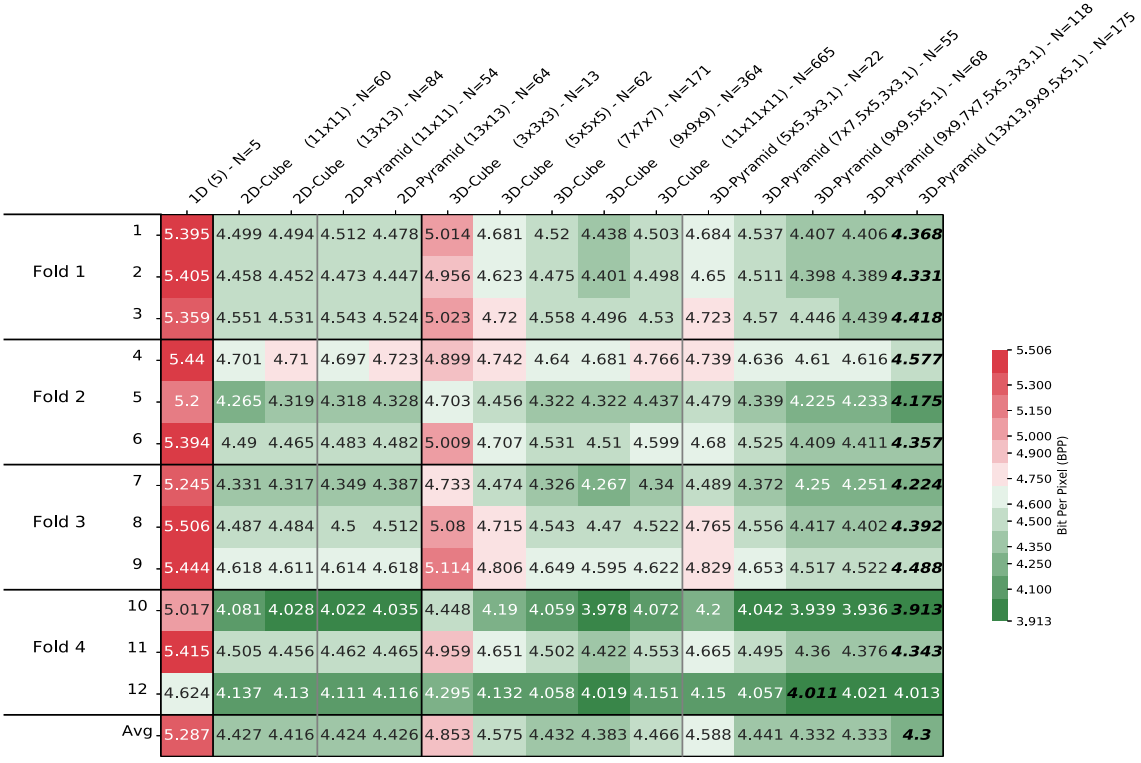



**Fig. 6.** Benchmark 1 result that illustrates the compression ratio in Bits-per-pixel (bpp) for compressing **Dataset1** using models trained on different neighbouring sequences applied in 4.3. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 3.913 bpp (Green) to the lowest compression 5.506 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

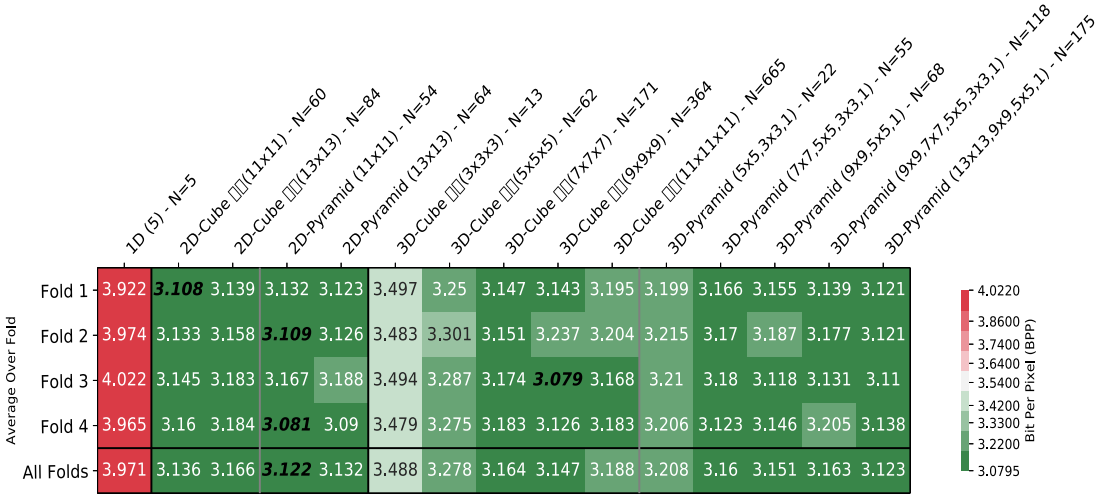



**Fig. 7.** Compression performance in (bpp) for evaluating the trained models with different neighbouring sequences from Benchmark 1 validated over **Dataset2**. Models pre-trained on distinct neighbouring sequences from **Dataset1** are evaluated over this unseen set while estimating the bpp mean pre fold. The overall average bpp achievements across all folds' means for each model is proposed in the last row. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 3.0795 bpp (Green) to the lowest compression 4.0220 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

### *4.2. Performance metrics*

4-fold cross-validation has been applied to evaluate the accuracy and performance of our LSTM predictive models for each of the different neighbouring sequence types over **Dataset1**. Each fold consists of nine volumes belonging to the training set and three for the testing. Generally, such a validation technique aims to measure a trained model's effectiveness and generalisation on unseen data. Moreover, additional experimental tests have been conducted over an out of domain public data **Dataset2** to further evaluate the generalisability and robustness of the trained models across other unseen and distinctive medical modalities (e.g. MRI).

The bits-per-pixel (bpp) (4) has been chosen to be the evaluation metric of the compression ratio obtained by all our LSTM models.

$$\text{bpp} = \frac{\text{Compressed Image Size (Bits)}}{\text{Number of Voxels}} \tag{4}$$

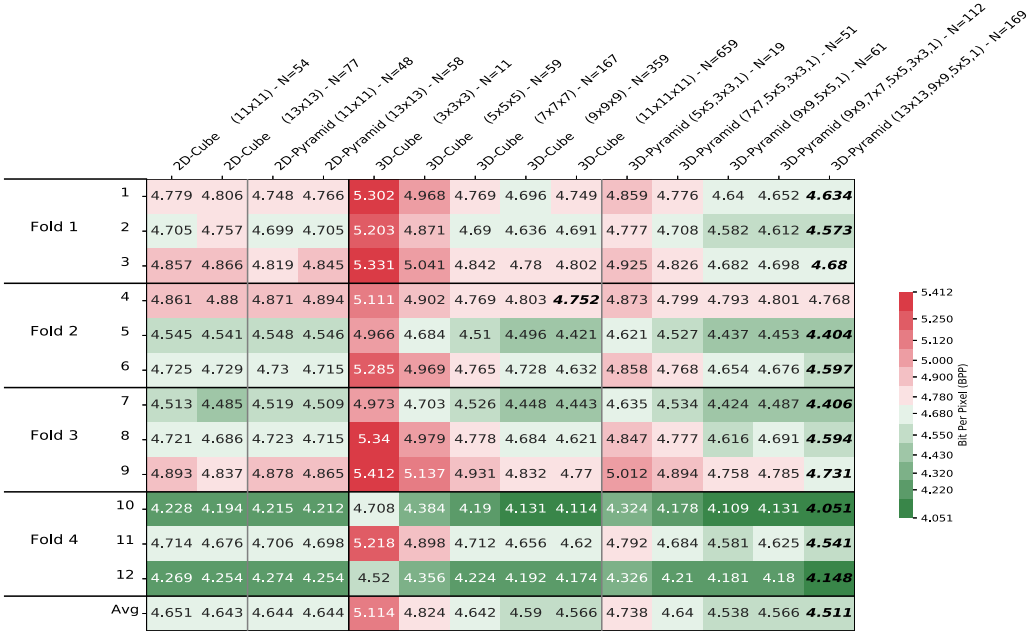



**Fig. 8.** Benchmark 2 result that illustrates the compression ratio in (bpp) for compressing **Dataset1** using models trained on **reduced** neighbourhood sequences applied in Section 4.4. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 4.051 bpp (Green) to the lowest compression 5.412 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

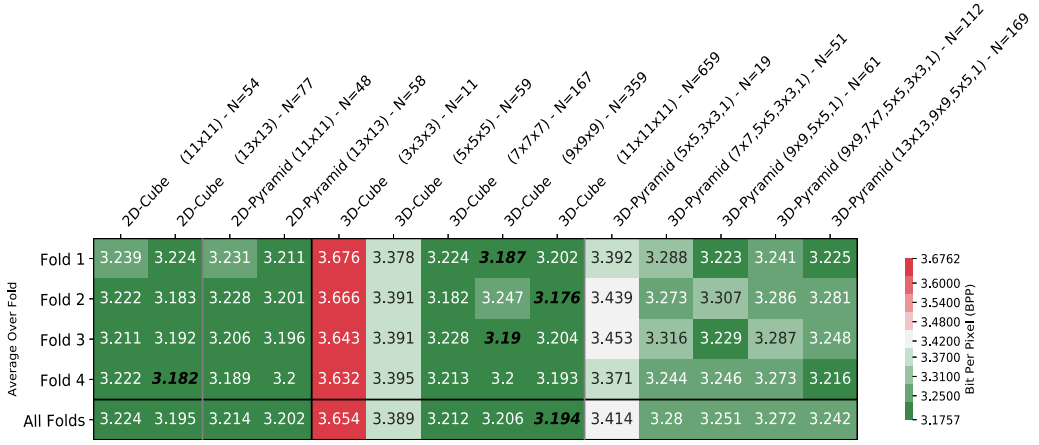



**Fig. 9.** Compression performance in (bpp) for evaluating the trained models with different neighbouring sequences from Benchmark 2 validated over **Dataset2**. Models pre-trained on the **reduced** input configurations from **Dataset1** are now evaluated over this unseen set **Dataset2** while estimating the bpp mean pre fold. The overall average bpp achievements across all folds' means for each model is proposed in the last row. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 3.1757 bpp (Green) to the lowest compression 3.6762 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Compression time (in seconds) is measured per model for both the encoding and the decoding operations.

### 4.3. Result for Benchmark 1: Optimal input sequence (shape and size)

This experiment determines the best input sequence (shape and size) that would lead to optimal compression for 3D medical images. Fig. 6 illustrates the compression ratio in bpp for each model trained on the neighbouring sequence options. Volumes are grouped into their respective cross-validation fold, wherein each row represents a volume and each fold contains validation over three volumes — separated with a horizontal black line. The last row reports the average bpp of models through all volumes. Each column represents a model trained on a specific neighbouring sequence case with the shape, size and sequence length indicated. Cells are coloured from maximum compression 3.913 bpp (Green) to the lowest compression 5.506 bpp (Red). The best compression result is in bold. The model trained on a 1D sequence produces the worst compression ratio because only five previous voxels from only the $x$-axis were utilised. The models trained on sequences extracted from a 2D slice gain better compression performance for both cube and triangle shapes. The 2D triangle sequences are more promising since they use fewer neighbouring voxels but still gain comparable compression results to the 2D block cases. For 3D sequences with cubic shape, as the block size increases, those models' compression performance improves apart for block size ($11^3$), which may need longer training or a larger model. Within the 3D cube models, the model trained on input vector with ($3^3$) block size produces the least compression ratio while the model trained on a block size of ($9^3$) gains the best compression. LSTM models trained on sequences with a 3D pyramid shape achieve an excellent compression performance, starting

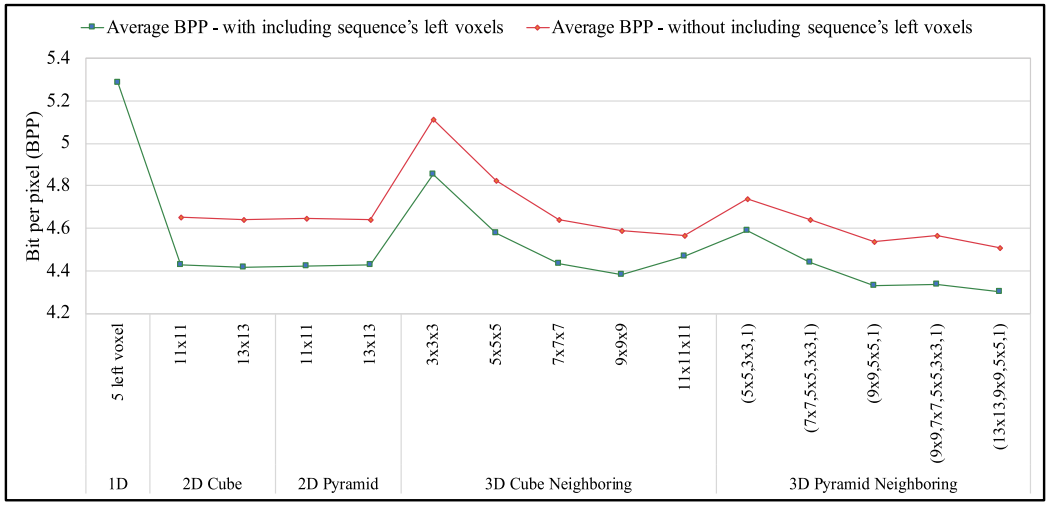



**Fig. 10.** A summary overview of the average (bpp) over Dataset1's volumes using all models trained with left voxels (in Green) and without (in Red) applied in Benchmark 1, and Benchmark 2, respectively. Overall, models pre-trained on reduced inputs result in a negligible performance drop of $\approx 0.2$ bpp, but with a significant positive impact on the compression times (see Section 4.5 for further details). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

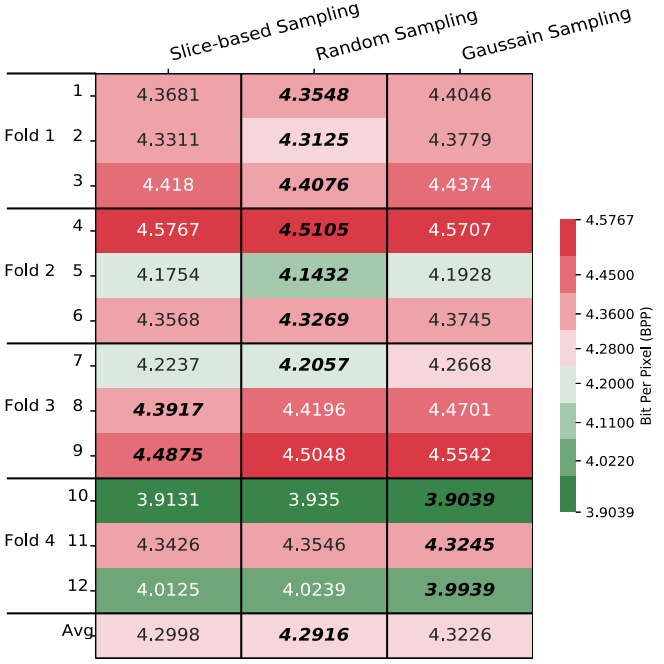



**Fig. 11.** Benchmark 4 results, which empirically demonstrates the bits-per-pixel (bpp) for each sampling strategy whereby training batches are uniquely extracted from the 3D CT scans in **Dataset1**. Cells are highlighted from maximum compression 3.9039 bpp (Green) to minimum compression 4.5767 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

with the smallest block size with just 22 voxels. This input vector allows its LSTM model to gain comparable compression result to the one trained on the 3D cube with $N = 62$ voxels. The following block size with an input length of $N = 55$, obtains a compression reduction similar to the cubic input with N=171 voxels. The next three models with surrounding neighbouring sequences gain the best compression among all the other options. Interestingly, it appears that models trained on the 3D pyramid with $(9 \times 9, 5 \times 5, 1)$ and $(13 \times 13, 9 \times 9, 5 \times 5, 1)$ accomplish this result due to involving more voxels from $x$ and $y$-axis while still maintaining information form the depth slice $z$-axis (pyramid with wider base). Overall, the pyramid neighbouring sequence forms a good balance between utilising the contextual information around the target voxel while keeping the sequence length compact, thus allowing faster training.

Fig. 7 presents a validation of all trained models over **Dataset2** wherein models pre-trained on distinct input configurations from **Dataset1** are evaluated on this unseen set. Each row represents the average compression ratio of different models over a single fold, while each column illustrates the reduction ratios of four models trained on the same input pattern across different volumes. The last row estimates the overall average bpp achievements across all folds' means. Cells are coloured from maximum compression 3.0795 bpp (Green) to the lowest compression 4.0220 bpp (Red) —(The full validation results are proposed in Appendix B). Although our proposed models were not trained on this dataset, they still generalise well to all volumes belonging to this data. As expected, the model trained on the shortest input sequence (i.e. 1D) has the worst compression ratio with a bpp ratio of 3.971 on average. Across this particular dataset, it appears that the models trained on 2D patterns for both cube and triangle shapes gain comparable compression performance to some 3D input configurations. The performance gain in 2D cases is expected when recognising the similarity in pixel spacing quality to what the models were trained on (i.e. trained on $[.488, .488]$ and evaluated on $[.5, .5]$). While in the case of 3D quality, there is a significant variation in the slice thickness between what the models learned from (i.e. .625 mm) and validated on (i.e. 2 mm slice thickness). Models trained on input with 3D cube shapes gradually gain better compression ratios as their block size expand, starting from least performer with an average of 3.488 bpp to their highest compression result of 3.147 bpp on average. Compared to the 3D cube configurations, the 3D pyramid offers a more condensed inputs' length yet better compression results, achieving a mean of 3.123 bpp by the best compressor. Overall, the proposed models with 2D triangle $(11 \times 11)$, 3D Cube $(9^3)$ and 3D pyramid $(13 \times 13, 9 \times 9, 5 \times 5, 1)$ input sequences obtain the best bpp reductions over **Dataset2** compared to other competitors gaining 3.081 bpp, 3.079 bpp, and 3.11 bpp, respectively.

### 4.4. Result for Benchmark 2: Optimising decoder performance

Parallel decoding is enabled by introducing new sequences that drop the left neighbouring voxels. Fig. 8 presents the compression ratio in bpp for each of the new neighbouring sequence options. 4-fold cross-validation was applied over **Dataset1**. Each column represents model performance trained on a specific sequence. The last row presents the average bpp per model across all volumes. Cells are coloured from

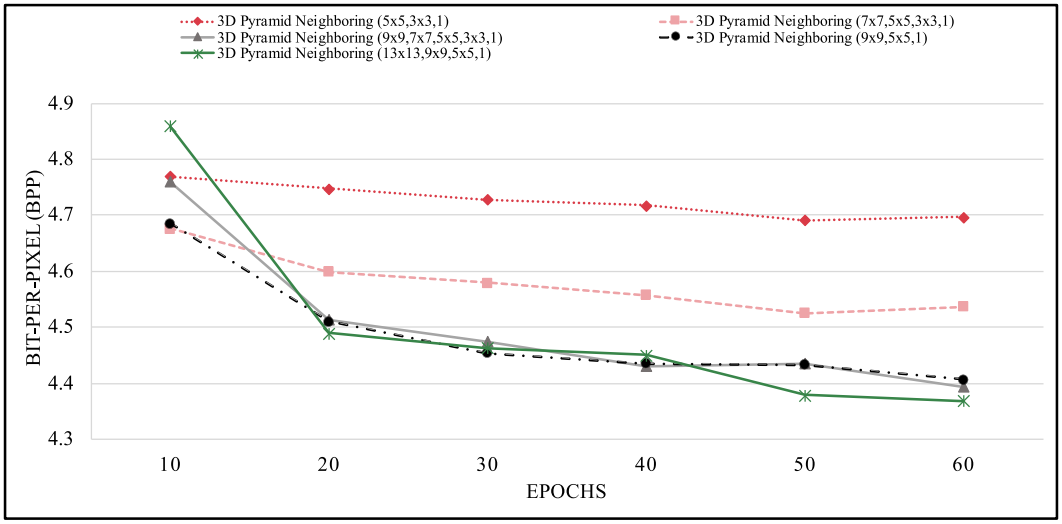



**Fig. 12.** Benchmark 5 result, which illustrates (bpp) variations during models' training steps. To clarify, this plot does not demonstrate the model's training loss function, but it evaluates trained models' compression qualities after different epochs. Models with pyramid input vectors (including left voxels from Benchmark 1) are evaluated after 10, 20, 30, 40, 50 and 60 epochs.

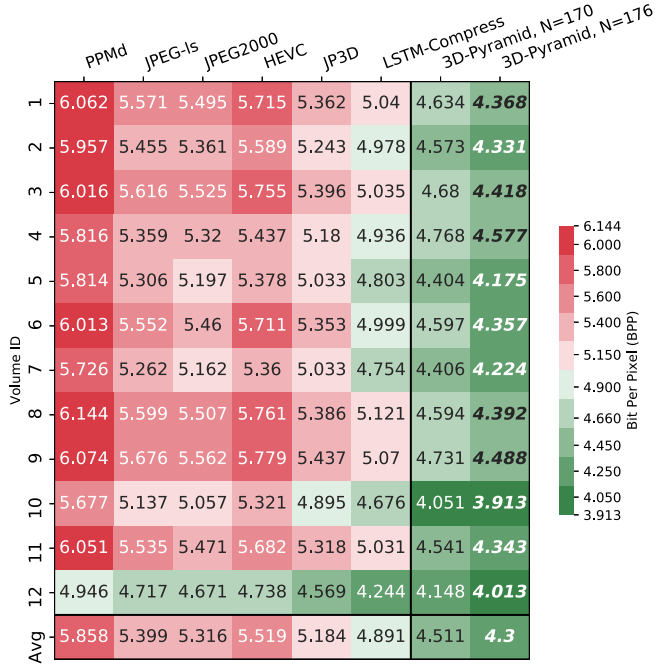



**Fig. 13.** Benchmark 6 result that illustrates the compression ratio in bpp for two of the proposed models compared the state-of-the-art lossless compression methods over **Dataset1** (16-bits volumes). Cells are highlighted from maximum compression 3.913 bpp (Green) to minimum compression 6.144 bpp (red). Our two proposed models have 3D pyramid shapes with $(13\times13, 9\times9, 5\times5, 1)$ while $N = 170$ forms sequence without including left voxels, and $N = 176$ is sequence including left voxels. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

maximum compression 4.051 bpp (Green) to least 5.412 bpp (Red). The best compression result is in bold. In this benchmark, the 1D input sequence is not applicable. Models trained on 2D neighbourhood sequences all produced similar results, with the triangle-shaped sequences having lower costs since fewer voxels are used. Notably, training on cubic-shaped input demonstrates better compression as block size increases. For instance, the model trained on $(3^3)$ has the least compression of 5.114 bpp, while the $(11^3)$ block size demonstrates a reduction of 4.566 bpp. However, when practically compared to models trained on sequences based on a 3D pyramid, higher reduction achievements are gained regardless of block size. In particular, the sequences with $N = 61$ and $N = 169$ voxels produce the best bit reduction among all the models over **Dataset1**. Overall, this novel input strategy's performance has a negligible drop of $\approx 0.2$ bpp in compression ratio compared to the non-parallel version (Fig. 6), which is expected when removing some of the contextual information from the sequences. However, this comes with a significant positive impact on the compression times (see next Section 4.5). A summary overview of the average storage impact of the method bpp over all 12 volumes for the two experiments is presented in Fig. 10.

In Fig. 9, models trained on the reduced input configurations from different folds of **Dataset1** are evaluated in bpp over this out of domain public set (i.e. **Dataset2**). The overall average bpp achievements across all folds' means are proposed in the last row. Cells are coloured from maximum compression 3.1757 bpp (Green) to the lowest compression 3.6762 bpp (Red) —(The full validation results are proposed in Appendix B). Models pre-trained on 2D sequence formats all provide relatively similar compression results around 3.202 bpp that matched other 3D configurations across this specific dataset. The performance difference in the 3D models' cases is expected due to the variations in scanning quality, precisely the slice thickness between the training data (i.e. **Dataset1** .625 mm) and the testing data (i.e. **Dataset2** 2 mm). However, it is noticeable that many of the 3D input sequences still achieve a significant compression performance, for instance, 3.176 bpp reduction by the best performer. Within the models trained on 3D cube sequences, both $N = 359$ and $N = 659$ voxels options yield the best bit reductions with 3.187 bpp and 3.176 bpp. However, when recognising the compression time affected by their sequence lengths, the balance of compactness and speed of models with 3D pyramid inputs $N = 61$ and $N = 169$ is more desirable, obtaining comparable reductions with 3.216 bpp and 3.223 bpp. For **Dataset2**, the performance drop affected by the novel version of input formats is negligible, only $\approx .11$ bpp. Moreover, this comes at the significant achievement of the encoding–decoding speed-up enabled by parallel operation.

### 4.5. Result for Benchmark 3: Encoding-decoding performance

Table 1, presents the time in seconds (s) required to compress and decompress a single slice belong to **Dataset1** with all the different

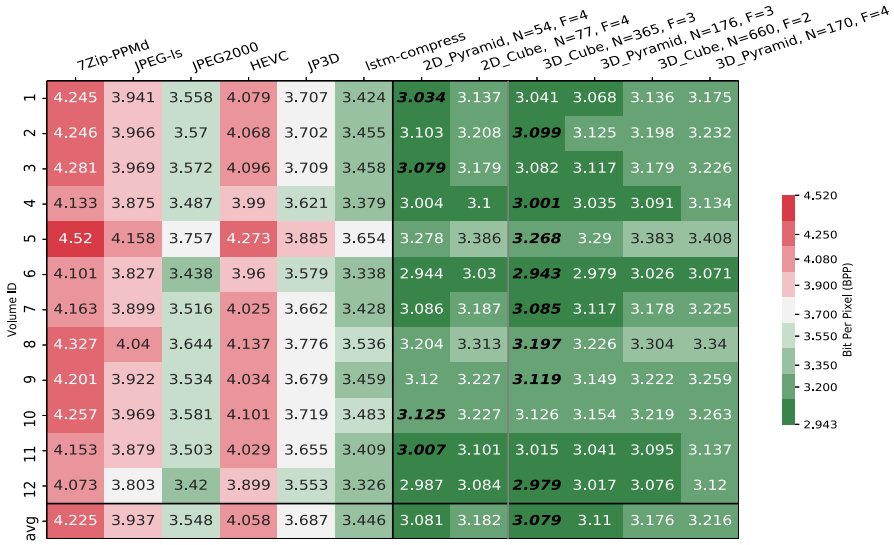



**Fig. 14.** Illustrating the compression ratio in bpp for evaluating some of our pre-trained models compared to the state-of-the-art lossless compression methods over **Dataset2** (16-bits volumes). Cells are highlighted from maximum compression 2.943 bpp (Green) to minimum compression 4.520 bpp (red). The proposed models are selections of best performers from different folds (emphasised by "F") and include models evaluated on distinctive 2D and 3D sequences with and without including the left voxels (emphasised by sequence's length "N").

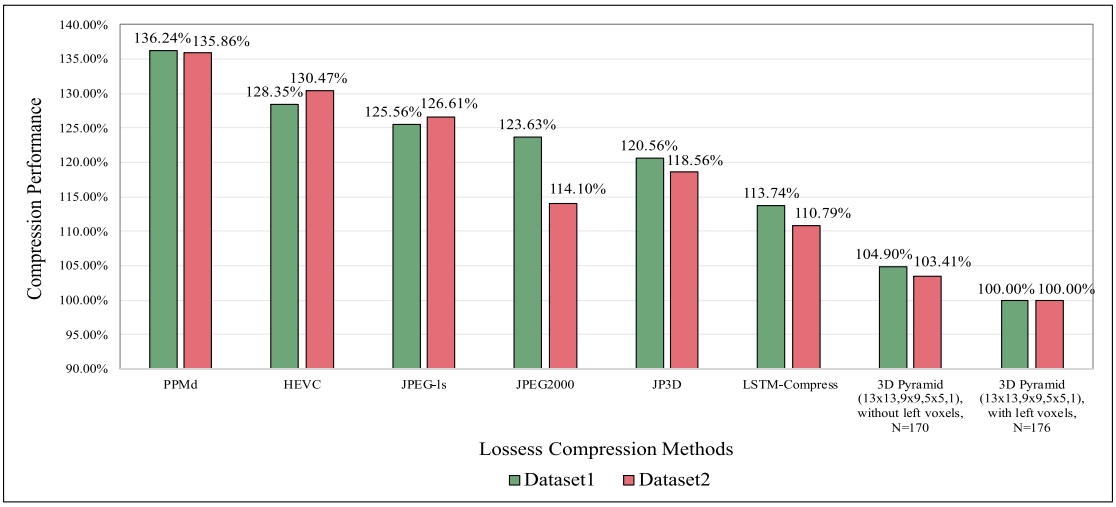



**Fig. 15.** A summary overview of the compression performance over different 3D medical datasets (16-bits) (**Dataset1** and **Dataset2**) using the proposed input sequences of LSTM predictor models compared to the state-of-the-art lossless compression methods (Less value indicates better performance).

neighbourhood models. The input length naturally influences the computation cost. The table compares the compression time using models trained with left voxels and the novel (reduced version) without left $x$-axis voxels. The model with the best bpp on average, is in bold. Although the compression ratio of models that include left voxels gain a better bpp reduction, the decoding time is significantly slower than encoding time regardless of the sequence specification. However, when comparing the models that remove the left voxels, the impact of leveraging parallelism is more noticeable with the same computation time for both encoding and decoding. By observing the results of Table 1, one concludes that the 3D pyramids with $(13\times13, 9\times9, 5\times5, 1)$, $N = 169$, and $(9\times9, 5\times5, 1)$, $N = 61$ present the best balance of compression time and ratio (5.95s, 4.511 bpp), and (3.6s, 4.538 bpp), respectively.

**Comparison to existing method:** We also evaluated against a comparative existing and available deep learning method (LSTM-Compress (Knoll, 2020)) for compressing the same slice. LSTM-Compress performed compression time and rate of (133.22s, 4.891 bpp), and (132.32s) for decompression time. Our two proposed models have noticeable compression reductions of (0.38 bpp, or 8.4%), and (0.353 bpp, or 7.8%), respectively, when comparing them against LSTM-Compress. Moreover, our proposed predictive models outperform LSTM-Compress producing speedup gains about 22× and 37× faster encoding–decoding performance.

### 4.6. Result for Benchmark 4: Sampling strategy

Fig. 11 shows the bpp of models trained on samples extracted by each sampling scheme from **Dataset1**. 4-fold cross-validation was applied to each method. The last row is the average bpp overall. Based on the experiment, all three strategies produce similar compression results on average. Overall, both random sampling and the slice-based scheme achieved the best reduction on average. It appears that training the model on whole slices with every voxel within the selected slices being available leads to improvement in the compression results and yields comparable results to the uniform sampling.

**Table 1**
Benchmark 3: encoding–decoding performance measured by (bits-per-pixel bpp, and time in seconds) for compressing and decompressing a single slice belongs to Dataset1. The comparison includes all models trained on different casual neighbouring sequences (with and without the left voxels) from Benchmark 1 and Benchmark 2. The best compression result (bpp) on average, is in bold.

| Neighbouring Sequence (Shape & Size) | Models trained **with including** sequence's left voxels | | | | Models trained **without including** sequence's left voxels | | | |
|---|---|---|---|---|---|---|---|---|
| | Sequence Length (l) | Average (BPP) | Compression Time (s) | Decompression Time (s) | Sequence Length (l) | Average (BPP) | Compression Time (s) | Decompression Time (s) |
| 1D (5) | 5 | 5.287 | 1.93 | 606.18 | – | – | – | – |
| 2D Block (11 × 11) | 60 | 4.427 | 3.19 | 1127.65 | 54 | 4.651 | 3.12 | 3.21 |
| 2D Block (13 × 13) | 84 | 4.416 | 3.73 | 1408.7 | 77 | 4.643 | 3.71 | 3.66 |
| 2D Triangle (11 × 11) | 54 | 4.424 | 2.96 | 1066.61 | 48 | 4.644 | 2.89 | 3.07 |
| 2D Triangle (13 × 13) | 64 | 4.426 | 3.17 | 1158.08 | 58 | 4.644 | 3.05 | 3.11 |
| 3D Cube (3 × 3 × 3) | 13 | 4.853 | 2.33 | 627.91 | 11 | 5.114 | 2.09 | 2.16 |
| 3D Cube (5 × 5 × 5) | 62 | 4.575 | 3.25 | 1167.93 | 59 | 4.824 | 3.22 | 3.56 |
| 3D Cube (7 × 7 × 7) | 171 | 4.432 | 5.76 | 2445.81 | 167 | 4.642 | 5.72 | 5.86 |
| 3D Cube (9 × 9 × 9) | 364 | 4.383 | 10.46 | 4776.11 | 359 | 4.59 | 10.24 | 10.44 |
| 3D Cube (11 × 11 × 11) | 665 | 4.466 | 17.6 | 8384.63 | 659 | 4.566 | 17.61 | 17.67 |
| 3D Pyramid (5 × 5,3 × 3,1) | 22 | 4.588 | 2.48 | 718.38 | 19 | 4.738 | 2.41 | 2.47 |
| 3D Pyramid (7 × 7,5 × 5,3 × 3,1) | 55 | 4.441 | 3.39 | 1049.63 | 51 | 4.64 | 3.29 | 3.34 |
| 3D Pyramid (9 × 9,5 × 5,1) | 68 | 4.332 | 3.5 | 1200.53 | 61 | 4.538 | 3.36 | 3.5 |
| 3D Pyramid (9 × 9,7 × 7,5 × 5,3 × 3,1) | 118 | 4.333 | 4.9 | 1815.33 | 112 | 4.566 | 4.61 | 4.69 |
| 3D Pyramid (13 × 13,9 × 9,5 × 5,1) | 175 | **4.3** | 6.14 | 2512.65 | 169 | **4.511** | 5.94 | 5.85 |
| Average | – | 4.512 | 4.986 | 2004.409 | – | 4.665 | 5.09 | 5.185 |

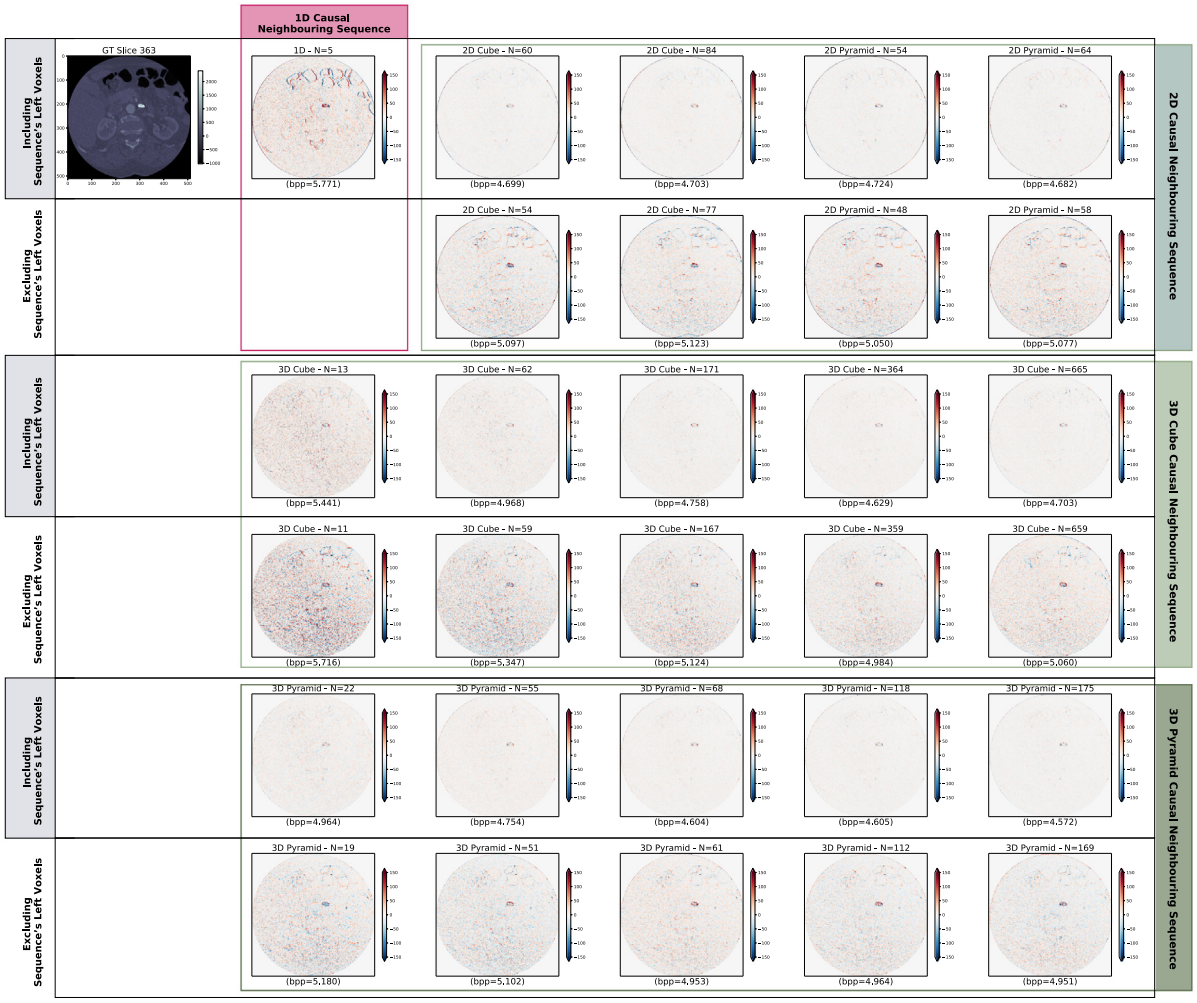



**Fig. A.1.** Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 1.

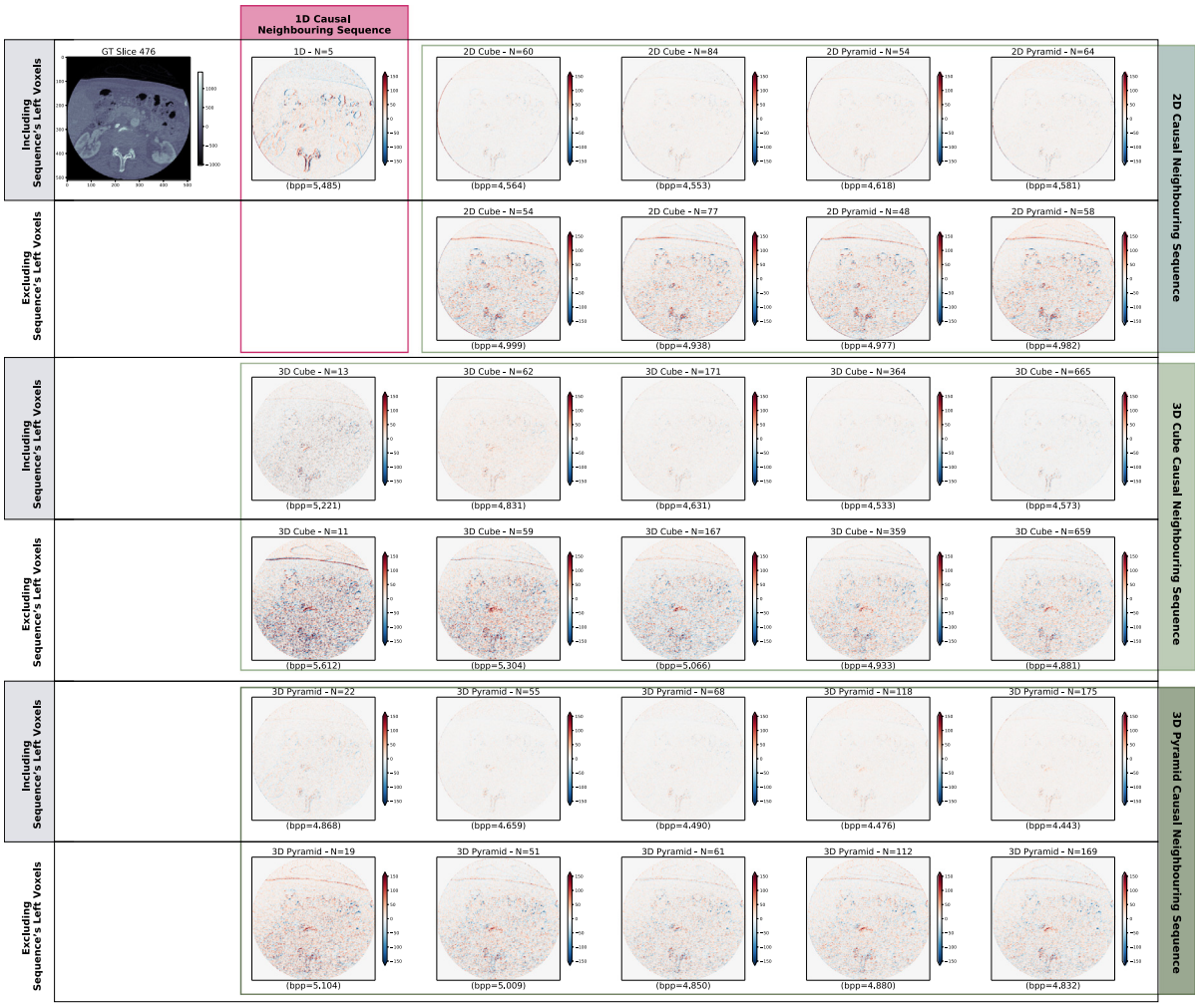



**Fig. A.2.** Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 7.

### *4.7. Result for Benchmark 5: Compression improvement during training*

Fig. 12 illustrates the change in (bpp) over-training epochs when compressing the same volume. This experiment shows the increase in the compression ratio when training models with different neighbourhood sequences, namely, the pyramid input vectors (including left voxels). To clarify, this plot does not illustrate each model's training loss function, but it evaluates trained models' compression bpp after different epochs. Among these cases, the pyramid with $(13\times13, 9\times9, 5\times5, 1)$ block size obtains the best reduction followed by $(9\times9, 7\times7, 5\times5, 1)$, and $(9\times9, 5\times5, 1)$

### *4.8. Result for Benchmark 6: Comparing with state-of-the-art lossless compression methods*

The experiment in Figs. 13 and 14 evaluate the compression performance in bpp for compressing **Dataset1** and **Dataset2** using the proposed models and some state-of-the-art lossless compression methods, including well-known image and video coders, namely, PPMd (Pavlov, 2019), JPEG-LS (e.V. OFFIS, 2020), JPEG2000 (OpenJPEG, 2019), HEVC (Bossen et al., 2019; Flynn, et al., 2016), JP3D (OpenJPEG, 2019), and the deep learning method LSTM-Compress (Knoll, 2020).

Among the standard codecs (Fig. 13), JP3D gains the best reduction over **Dataset1** with 5.184 bpp on average, followed by JPEG2000 with 5.316 bpp. Compared to the LSTM-Compress model, our two LSTM predictive-based models obtain 0.38 bpp and 0.591 bpp (or 8.84% and 13%) better reductions, respectively.

In Fig. 14, the compression performance over **Dataset2** is presented for each of the lossless compressors, including some well-known classical codecs, a deep learning alternative codec (i.e. LSTM-Compress), and our best model performers. Among the standard codecs, JPEG2000 outperforms other classical codecs with a compression performance of 3.548 bpp. Our two state-of-the-art models trained on input sequences with 3D pyramid-shapes save 10% and 7% compared to the competing LSTM-Compress codec. The same figure demonstrates the trade-offs between our individual proposed models in terms of compression performance and speed affected by choice of input sequences' shape and length options.

Overall, our one-step-ahead prediction models achieve state-of-the-art compression for all datasets with 17%, and 13% space saving over **Dataset1**, and 12%, and 9% storage improvement over **Dataset2** compared to the best performers among classical methods JP3D, and JPEG2000, respectively as illustrated in Fig. 15.

## 5. Conclusion

We present a thorough study of a supervised LSTM prediction-based model for compressing 3D medical images (12-bits data stored as 16-bits depth volumes) losslessly. Six main benchmarks were conducted to determine the optimal input sequence for medical domain compression. We evaluated all experiments by choosing the compression performance (bpp and time in seconds) as primary evaluation metrics. Moreover, many sequences of the causal neighbourhood were empirically investigated and analysed to highlight trade-offs. Furthermore,

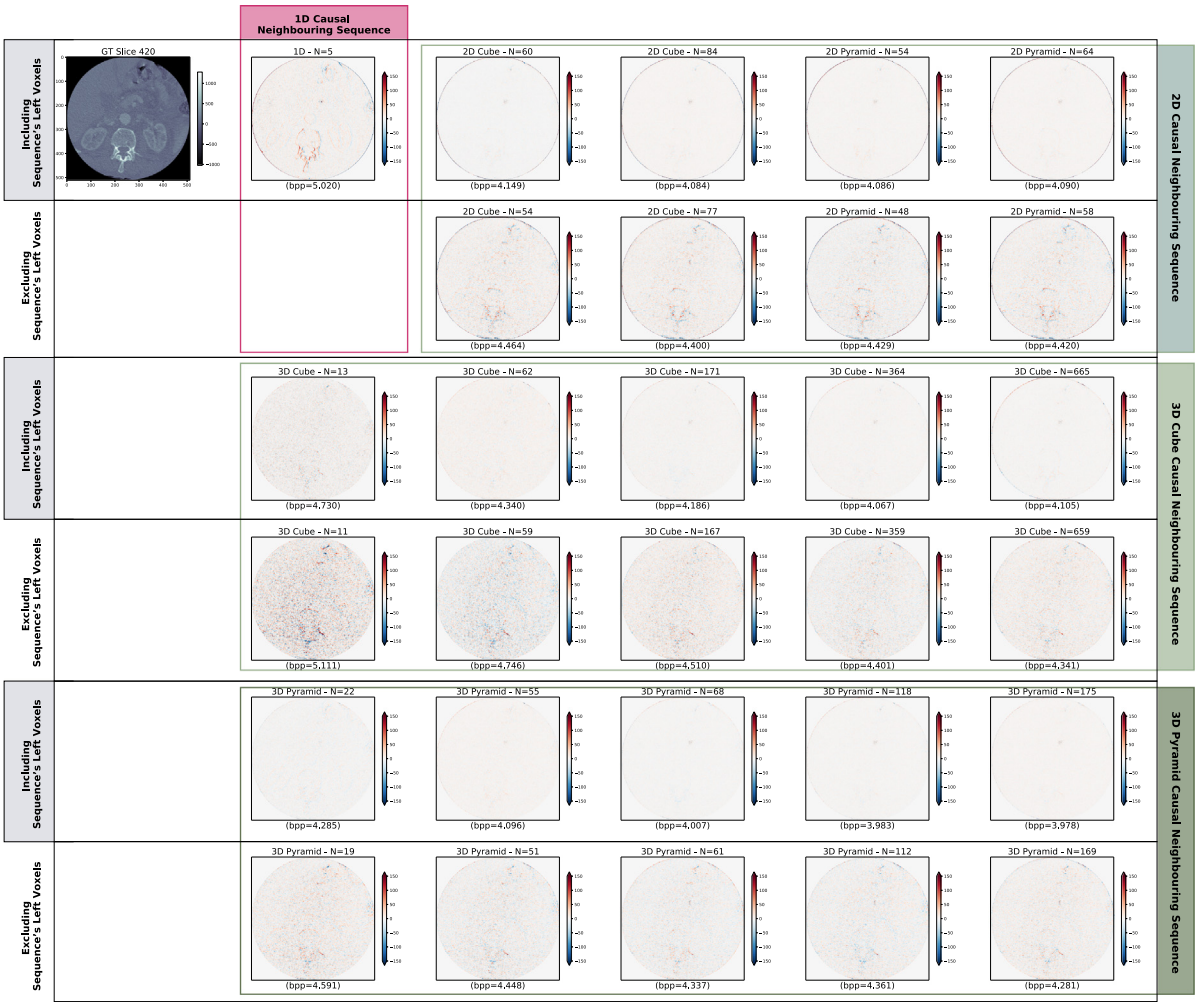



**Fig. A.3.** Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 10.

a novel and efficient type of input sequence was introduced (without immediate left voxels), which allows simple parallelism of the decoder and still provides a favourable compression. From the experimental results; we conclude that the pyramid-shaped input sequences accomplish state-of-the-art compression results compared to the other options and with a compression gain of $\approx 17\%$ and $\approx 12\%$ compared to the classical lossless compression alternatives over **Dataset1**, and **Dataset2**, respectively. Its effectiveness is driven by its balance between compactness and representativity, reflecting the local correlation around a target voxel. Moreover, the proposed reduced sequences reach almost a $3\times$ lossless compression ratio of the original volumes in the best case, and up to $500\times$ decoding speed-up can be achieved with little effect on the compression performance. Furthermore, the proposed trained models outperform other state-of-the-art lossless codecs in compressing all 3D medical volumes and for various modality types, including CT and MRI. In the future, we plan to extend this work to include different types of deep learning NN models for a predictive-based compression domain. Another promising research direction is investigating deep learning models' generalisation across other high-dimensional domain applications such as video.

## Declaration of competing interest

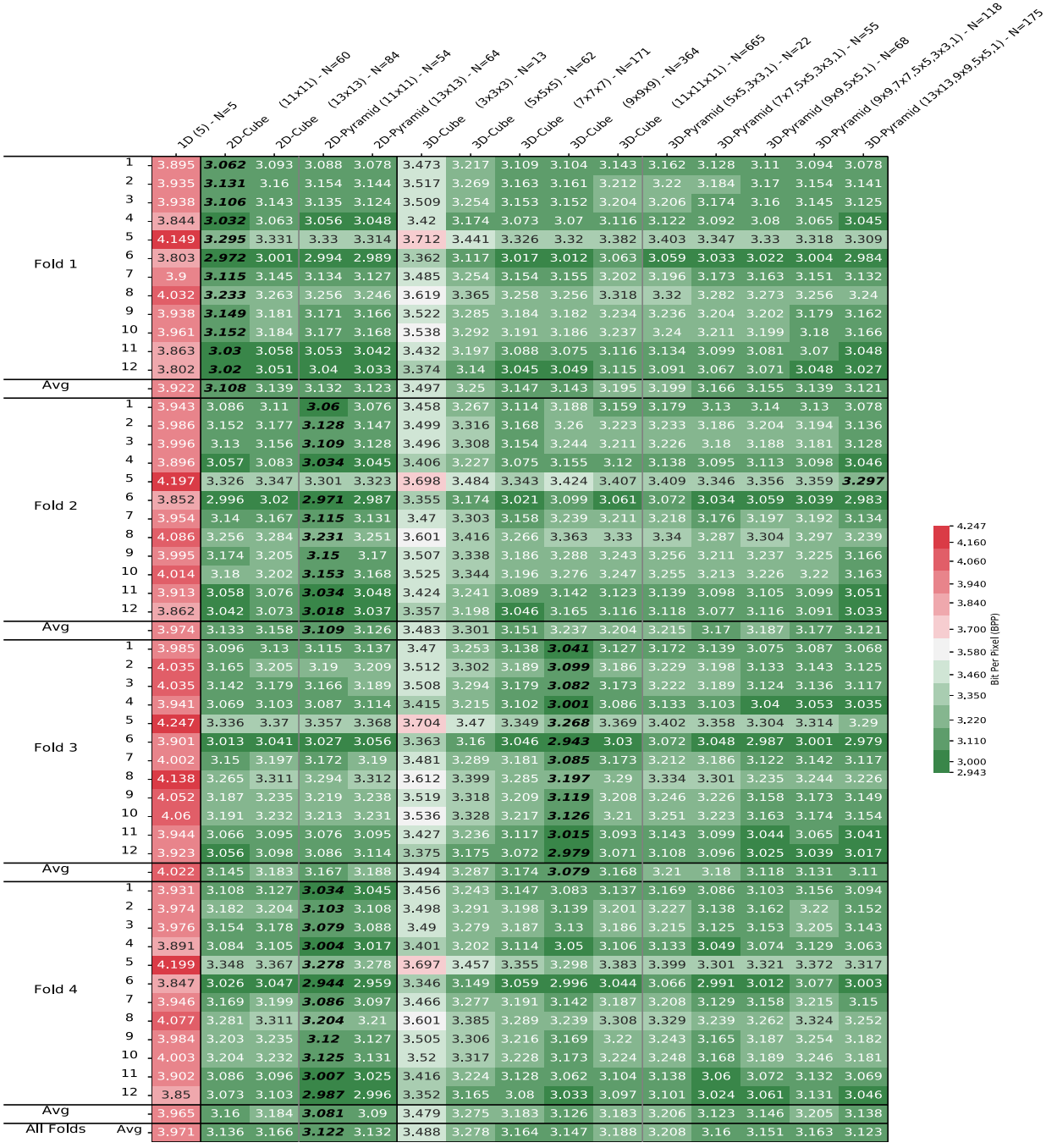
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Appendix A. Residual plots

In this section, we illustrate a number of residual comparisons plots from three random volumes. Each figure illustrates a plot of a mid-slice residual from one volume. It is only this residual slice that needs to be compressed using arithmetic coding (or similar technique) in order to provide lossless compression of the complete slice/dataset. All figures contain comparisons among the different casual neighbouring sequences categorised based on the sampling grid dimensions (e.g. 1D, 2D, and 3D). The odd rows of each plot are models trained on the sequence's **including** left voxels while the even rows are models trained **excluding** the sequence's left voxels. The first column illustrates the ground truth slice while the rest are plotting of the residual slices. Each sub-figure is titled with the input sequence's specifications including the sampling grid dimension and the sequence length. The compression ratio in (bpp) for each residual slice is included under each sub-figure. All residual slices are colourmaped with diverging colourmap, wherein the zero values are coloured with White, values less than zero are coloured with Blue, and values larger than zero are coloured with Red. Fig. A.1, A.2, and A.3 illustrates plots of middle residual slices selected randomly from volume one, seven, and ten, respectively.

## Appendix B. Validation on Dataset2

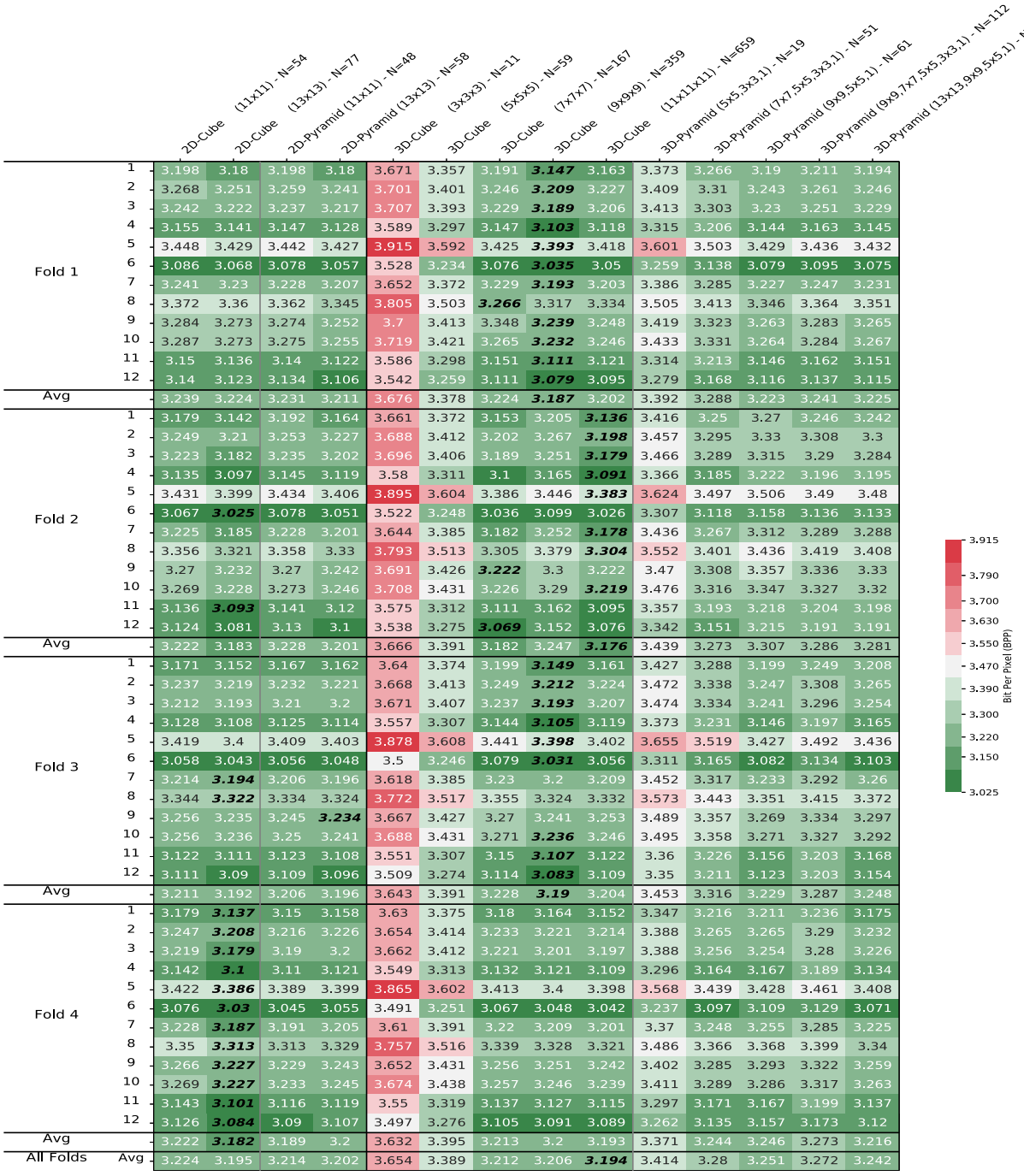
In this section, the full validation results of all models in Benchmark 1 and Benchmark 2 are illustrated in Fig. B.1 and Fig. B.2, respectively. In both figures, models pre-trained on distinct input configurations




| | | 1D (5) - N=5 | 2D-Cube (11x11) - N=60 | 2D-Cube (13x13) - N=84 | 2D-Pyramid (11x11) - N=54 | 2D-Pyramid (13x13) - N=64 | 3D-Cube (3x3x3) - N=13 | 3D-Cube (5x5x5) - N=62 | 3D-Cube (7x7x7) - N=171 | 3D-Cube (9x9x9) - N=364 | 3D-Pyramid (11x11x11) - N=665 | 3D-Pyramid (5x5,3x3,1) - N=22 | 3D-Pyramid (7x7,5x5,3x3,1) - N=55 | 3D-Pyramid (9x9,5x5,1) - N=68 | 3D-Pyramid (9x9,7x7,5x5,3x3,1) - N=118 | 3D-Pyramid (13x13,9x9,5x5,1) - N=175 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fold 1 | 1 | 3.895 | ***3.062*** | 3.093 | 3.088 | 3.078 | 3.473 | 3.217 | 3.109 | 3.104 | 3.143 | 3.162 | 3.128 | 3.11 | 3.094 | 3.078 |
| | 2 | 3.935 | ***3.131*** | 3.16 | 3.154 | 3.144 | 3.517 | 3.269 | 3.163 | 3.161 | 3.212 | 3.22 | 3.184 | 3.17 | 3.154 | 3.141 |
| | 3 | 3.938 | ***3.106*** | 3.143 | 3.135 | 3.124 | 3.509 | 3.254 | 3.153 | 3.152 | 3.204 | 3.206 | 3.174 | 3.16 | 3.145 | 3.125 |
| | 4 | 3.844 | ***3.032*** | 3.063 | 3.056 | 3.048 | 3.42 | 3.174 | 3.073 | 3.07 | 3.116 | 3.122 | 3.092 | 3.08 | 3.065 | 3.045 |
| | 5 | 4.149 | ***3.295*** | 3.331 | 3.33 | 3.314 | 3.712 | 3.441 | 3.326 | 3.32 | 3.382 | 3.403 | 3.347 | 3.33 | 3.318 | 3.309 |
| | 6 | 3.803 | ***2.972*** | 3.001 | 2.994 | 2.989 | 3.362 | 3.117 | 3.017 | 3.012 | 3.063 | 3.059 | 3.033 | 3.022 | 3.004 | 2.984 |
| | 7 | 3.9 | ***3.115*** | 3.145 | 3.134 | 3.127 | 3.485 | 3.254 | 3.154 | 3.155 | 3.202 | 3.196 | 3.173 | 3.163 | 3.151 | 3.132 |
| | 8 | 4.032 | ***3.233*** | 3.263 | 3.256 | 3.246 | 3.619 | 3.365 | 3.258 | 3.256 | 3.318 | 3.32 | 3.282 | 3.273 | 3.256 | 3.24 |
| | 9 | 3.938 | ***3.149*** | 3.181 | 3.171 | 3.166 | 3.522 | 3.285 | 3.184 | 3.182 | 3.234 | 3.236 | 3.204 | 3.202 | 3.179 | 3.162 |
| | 10 | 3.961 | ***3.152*** | 3.184 | 3.177 | 3.168 | 3.538 | 3.292 | 3.191 | 3.186 | 3.237 | 3.24 | 3.211 | 3.199 | 3.18 | 3.166 |
| | 11 | 3.863 | ***3.03*** | 3.058 | 3.053 | 3.042 | 3.432 | 3.197 | 3.088 | 3.075 | 3.116 | 3.134 | 3.099 | 3.081 | 3.07 | 3.048 |
| | 12 | 3.802 | ***3.02*** | 3.051 | 3.04 | 3.033 | 3.374 | 3.14 | 3.045 | 3.049 | 3.115 | 3.091 | 3.067 | 3.071 | 3.048 | 3.027 |
| | Avg | 3.922 | ***3.108*** | 3.139 | 3.132 | 3.123 | 3.497 | 3.25 | 3.147 | 3.143 | 3.195 | 3.199 | 3.166 | 3.155 | 3.139 | 3.121 |
| Fold 2 | 1 | 3.943 | 3.086 | 3.11 | ***3.06*** | 3.076 | 3.458 | 3.267 | 3.114 | 3.188 | 3.159 | 3.179 | 3.13 | 3.14 | 3.13 | 3.078 |
| | 2 | 3.986 | 3.152 | 3.177 | ***3.128*** | 3.147 | 3.499 | 3.316 | 3.168 | 3.26 | 3.223 | 3.233 | 3.186 | 3.204 | 3.194 | 3.136 |
| | 3 | 3.996 | 3.13 | 3.156 | ***3.109*** | 3.128 | 3.496 | 3.308 | 3.154 | 3.244 | 3.211 | 3.226 | 3.18 | 3.188 | 3.181 | 3.128 |
| | 4 | 3.896 | 3.057 | 3.083 | ***3.034*** | 3.045 | 3.406 | 3.227 | 3.075 | 3.155 | 3.12 | 3.138 | 3.095 | 3.113 | 3.098 | 3.046 |
| | 5 | 4.197 | 3.326 | 3.347 | 3.301 | 3.323 | 3.698 | 3.484 | 3.343 | 3.424 | 3.407 | 3.409 | 3.346 | 3.356 | 3.359 | ***3.297*** |
| | 6 | 3.852 | 2.996 | 3.02 | ***2.971*** | 2.987 | 3.355 | 3.174 | 3.021 | 3.099 | 3.061 | 3.072 | 3.034 | 3.059 | 3.039 | 2.983 |
| | 7 | 3.954 | 3.14 | 3.167 | ***3.115*** | 3.131 | 3.47 | 3.303 | 3.158 | 3.239 | 3.211 | 3.218 | 3.176 | 3.197 | 3.192 | 3.134 |
| | 8 | 4.086 | 3.256 | 3.284 | ***3.231*** | 3.251 | 3.601 | 3.416 | 3.266 | 3.363 | 3.33 | 3.34 | 3.287 | 3.304 | 3.297 | 3.239 |
| | 9 | 3.995 | 3.174 | 3.205 | ***3.15*** | 3.17 | 3.507 | 3.338 | 3.186 | 3.288 | 3.243 | 3.256 | 3.211 | 3.237 | 3.225 | 3.166 |
| | 10 | 4.014 | 3.18 | 3.202 | ***3.153*** | 3.168 | 3.525 | 3.344 | 3.196 | 3.276 | 3.247 | 3.255 | 3.213 | 3.226 | 3.22 | 3.163 |
| | 11 | 3.913 | 3.058 | 3.076 | ***3.034*** | 3.048 | 3.424 | 3.241 | 3.089 | 3.142 | 3.123 | 3.139 | 3.098 | 3.105 | 3.099 | 3.051 |
| | 12 | 3.862 | 3.042 | 3.073 | ***3.018*** | 3.037 | 3.357 | 3.198 | 3.046 | 3.165 | 3.116 | 3.118 | 3.077 | 3.116 | 3.091 | 3.033 |
| | Avg | 3.974 | 3.133 | 3.158 | ***3.109*** | 3.126 | 3.483 | 3.301 | 3.151 | 3.237 | 3.204 | 3.215 | 3.17 | 3.187 | 3.177 | 3.121 |
| Fold 3 | 1 | 3.985 | 3.096 | 3.13 | 3.115 | 3.137 | 3.47 | 3.253 | 3.138 | ***3.041*** | 3.127 | 3.172 | 3.139 | 3.075 | 3.087 | 3.068 |
| | 2 | 4.035 | 3.165 | 3.205 | 3.19 | 3.209 | 3.512 | 3.302 | 3.189 | ***3.099*** | 3.186 | 3.229 | 3.198 | 3.133 | 3.143 | 3.125 |
| | 3 | 4.035 | 3.142 | 3.179 | 3.166 | 3.189 | 3.508 | 3.294 | 3.179 | ***3.082*** | 3.173 | 3.222 | 3.189 | 3.124 | 3.136 | 3.117 |
| | 4 | 3.941 | 3.069 | 3.103 | 3.087 | 3.114 | 3.415 | 3.215 | 3.102 | ***3.001*** | 3.086 | 3.133 | 3.103 | 3.04 | 3.053 | 3.035 |
| | 5 | 4.247 | 3.336 | 3.37 | 3.357 | 3.368 | 3.704 | 3.47 | 3.349 | ***3.268*** | 3.369 | 3.402 | 3.358 | 3.304 | 3.314 | 3.29 |
| | 6 | 3.901 | 3.013 | 3.041 | 3.027 | 3.056 | 3.363 | 3.16 | 3.046 | ***2.943*** | 3.03 | 3.072 | 3.048 | 2.987 | 3.001 | 2.979 |
| | 7 | 4.002 | 3.15 | 3.197 | 3.172 | 3.19 | 3.481 | 3.289 | 3.181 | ***3.085*** | 3.173 | 3.212 | 3.186 | 3.122 | 3.142 | 3.117 |
| | 8 | 4.138 | 3.265 | 3.311 | 3.294 | 3.312 | 3.612 | 3.399 | 3.285 | ***3.197*** | 3.29 | 3.334 | 3.301 | 3.235 | 3.244 | 3.226 |
| | 9 | 4.052 | 3.187 | 3.235 | 3.219 | 3.238 | 3.519 | 3.318 | 3.209 | ***3.119*** | 3.208 | 3.246 | 3.226 | 3.158 | 3.173 | 3.149 |
| | 10 | 4.06 | 3.191 | 3.232 | 3.213 | 3.231 | 3.536 | 3.328 | 3.217 | ***3.126*** | 3.21 | 3.251 | 3.223 | 3.163 | 3.174 | 3.154 |
| | 11 | 3.944 | 3.066 | 3.095 | 3.076 | 3.095 | 3.427 | 3.236 | 3.117 | ***3.015*** | 3.093 | 3.143 | 3.099 | 3.044 | 3.065 | 3.041 |
| | 12 | 3.923 | 3.056 | 3.098 | 3.086 | 3.114 | 3.375 | 3.175 | 3.072 | ***2.979*** | 3.071 | 3.108 | 3.096 | 3.025 | 3.039 | 3.017 |
| | Avg | 4.022 | 3.145 | 3.183 | 3.167 | 3.188 | 3.494 | 3.287 | 3.174 | ***3.079*** | 3.168 | 3.21 | 3.18 | 3.118 | 3.131 | 3.11 |
| Fold 4 | 1 | 3.931 | 3.108 | 3.127 | ***3.034*** | 3.045 | 3.456 | 3.243 | 3.147 | 3.083 | 3.137 | 3.169 | 3.086 | 3.103 | 3.156 | 3.094 |
| | 2 | 3.974 | 3.182 | 3.204 | ***3.103*** | 3.108 | 3.498 | 3.291 | 3.198 | 3.139 | 3.201 | 3.227 | 3.138 | 3.162 | 3.22 | 3.152 |
| | 3 | 3.976 | 3.154 | 3.178 | ***3.079*** | 3.088 | 3.49 | 3.279 | 3.187 | 3.13 | 3.186 | 3.215 | 3.125 | 3.153 | 3.205 | 3.143 |
| | 4 | 3.891 | 3.084 | 3.105 | ***3.004*** | 3.017 | 3.401 | 3.202 | 3.114 | 3.05 | 3.106 | 3.133 | 3.049 | 3.074 | 3.129 | 3.063 |
| | 5 | 4.199 | 3.348 | 3.367 | ***3.278*** | 3.278 | 3.697 | 3.457 | 3.355 | 3.298 | 3.383 | 3.399 | 3.301 | 3.321 | 3.372 | 3.317 |
| | 6 | 3.847 | 3.026 | 3.047 | ***2.944*** | 2.959 | 3.346 | 3.149 | 3.059 | 2.996 | 3.044 | 3.066 | 2.991 | 3.012 | 3.077 | 3.003 |
| | 7 | 3.946 | 3.169 | 3.199 | ***3.086*** | 3.097 | 3.466 | 3.277 | 3.191 | 3.142 | 3.187 | 3.208 | 3.129 | 3.158 | 3.215 | 3.15 |
| | 8 | 4.077 | 3.281 | 3.311 | ***3.204*** | 3.21 | 3.601 | 3.385 | 3.289 | 3.239 | 3.308 | 3.329 | 3.239 | 3.262 | 3.324 | 3.252 |
| | 9 | 3.984 | 3.203 | 3.235 | ***3.12*** | 3.127 | 3.505 | 3.306 | 3.216 | 3.169 | 3.22 | 3.243 | 3.165 | 3.187 | 3.254 | 3.182 |
| | 10 | 4.003 | 3.204 | 3.232 | ***3.125*** | 3.131 | 3.52 | 3.317 | 3.228 | 3.173 | 3.224 | 3.248 | 3.168 | 3.189 | 3.246 | 3.181 |
| | 11 | 3.902 | 3.086 | 3.096 | ***3.007*** | 3.025 | 3.416 | 3.224 | 3.128 | 3.062 | 3.104 | 3.138 | 3.06 | 3.072 | 3.132 | 3.069 |
| | 12 | 3.85 | 3.073 | 3.103 | ***2.987*** | 2.996 | 3.352 | 3.165 | 3.08 | 3.033 | 3.097 | 3.101 | 3.024 | 3.061 | 3.131 | 3.046 |
| | Avg | 3.965 | 3.16 | 3.184 | ***3.081*** | 3.09 | 3.479 | 3.275 | 3.183 | 3.126 | 3.183 | 3.206 | 3.123 | 3.146 | 3.205 | 3.138 |
| All Folds | Avg | 3.971 | 3.136 | 3.166 | ***3.122*** | 3.132 | 3.488 | 3.278 | 3.164 | 3.147 | 3.188 | 3.208 | 3.16 | 3.151 | 3.163 | 3.123 |

**Fig. B.1.** Compression performance in (bpp) for evaluating the trained models with different neighbouring sequences from Benchmark 1 Section 4.3 validated over **Dataset2**. Models pre-trained on distinct neighbouring sequences from **Dataset1** are evaluated over this unseen set and grouped into corresponding folds separated by horizontal lines. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 2.9432 bpp (Green) to the lowest compression 4.247 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

from **Dataset1** are evaluated on this out of domain public set **Dataset2** and grouped into corresponding folds separated by horizontal lines. Each row represents the compression ratio of different models over a single volume, while each column illustrates the reduction ratios of four models trained on the same input pattern across different volumes. The last row estimates the average bpp achievements overall folds' means. Overall, although our proposed models were not trained on this dataset, they still generalise well to all volumes belonging to this data. Fig. B.1 presents a validation of all trained models from Benchmark 1 over **Dataset2**. Cells are coloured from maximum compression 2.9432 bpp (Green) to the lowest compression 4.247 bpp (Red). On the other hand, Fig. B.2 shows models trained on the **reduced** input configurations from different folds of **Dataset1** when evaluating them in bpp over this unseen set represented of Benchmark 2. Cells are coloured from maximum compression 3.025 bpp (Green) to the lowest compression 3.915 bpp (Red). From observing the evaluation results over all folds, one may interestingly recognise that all pre-trained models, regardless of their input formats, accomplish the worst compression results on volume 5 while best compression reductions were gained on volume 6. The compression performance on these two volumes has similar results by the state-of-the-art compression methods.




| | | 2D-Cube (11x11) - N=54 | 2D-Cube (13x13) - N=77 | 2D-Pyramid (11x11) - N=48 | 2D-Pyramid (13x13) - N=58 | 3D-Cube (3x3x3) - N=11 | 3D-Cube (5x5x5) - N=59 | 3D-Cube (7x7x7) - N=167 | 3D-Cube (9x9x9) - N=359 | 3D-Cube (11x11x11) - N=659 | 3D-Pyramid (5x5,3x3,1) - N=19 | 3D-Pyramid (7x7,5x5,3x3,1) - N=51 | 3D-Pyramid (9x9,5x5,1) - N=61 | 3D-Pyramid (9x9,7x7,5x5,3x3,1) - N=112 | 3D-Pyramid (13x13,9x9,5x5,1) - N=169 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Fold 1 | 1 | 3.198 | 3.18 | 3.198 | 3.18 | 3.671 | 3.357 | 3.191 | **3.147** | 3.163 | 3.373 | 3.266 | 3.19 | 3.211 | 3.194 |
| | 2 | 3.268 | 3.251 | 3.259 | 3.241 | 3.701 | 3.401 | 3.246 | **3.209** | 3.227 | 3.409 | 3.31 | 3.243 | 3.261 | 3.246 |
| | 3 | 3.242 | 3.222 | 3.237 | 3.217 | 3.707 | 3.393 | 3.229 | **3.189** | 3.206 | 3.413 | 3.303 | 3.23 | 3.251 | 3.229 |
| | 4 | 3.155 | 3.141 | 3.147 | 3.128 | 3.589 | 3.297 | 3.147 | **3.103** | 3.118 | 3.315 | 3.206 | 3.144 | 3.163 | 3.145 |
| | 5 | 3.448 | 3.429 | 3.442 | 3.427 | 3.915 | 3.592 | 3.425 | **3.393** | 3.418 | 3.601 | 3.503 | 3.429 | 3.436 | 3.432 |
| | 6 | 3.086 | 3.068 | 3.078 | 3.057 | 3.528 | 3.234 | 3.076 | **3.035** | 3.05 | 3.259 | 3.138 | 3.079 | 3.095 | 3.075 |
| | 7 | 3.241 | 3.23 | 3.228 | 3.207 | 3.652 | 3.372 | 3.229 | **3.193** | 3.203 | 3.386 | 3.285 | 3.227 | 3.247 | 3.231 |
| | 8 | 3.372 | 3.36 | 3.362 | 3.345 | 3.805 | 3.503 | **3.266** | 3.317 | 3.334 | 3.505 | 3.413 | 3.346 | 3.364 | 3.351 |
| | 9 | 3.284 | 3.273 | 3.274 | 3.252 | 3.7 | 3.413 | 3.348 | **3.239** | 3.248 | 3.419 | 3.323 | 3.263 | 3.283 | 3.265 |
| | 10 | 3.287 | 3.273 | 3.275 | 3.255 | 3.719 | 3.421 | 3.265 | **3.232** | 3.246 | 3.433 | 3.331 | 3.264 | 3.284 | 3.267 |
| | 11 | 3.15 | 3.136 | 3.14 | 3.122 | 3.586 | 3.298 | 3.151 | **3.111** | 3.121 | 3.314 | 3.213 | 3.146 | 3.162 | 3.151 |
| | 12 | 3.14 | 3.123 | 3.134 | 3.106 | 3.542 | 3.259 | 3.111 | **3.079** | 3.095 | 3.279 | 3.168 | 3.116 | 3.137 | 3.115 |
| | Avg | 3.239 | 3.224 | 3.231 | 3.211 | 3.676 | 3.378 | 3.224 | **3.187** | 3.202 | 3.392 | 3.288 | 3.223 | 3.241 | 3.225 |
| Fold 2 | 1 | 3.179 | 3.142 | 3.192 | 3.164 | 3.661 | 3.372 | 3.153 | 3.205 | **3.136** | 3.416 | 3.25 | 3.27 | 3.246 | 3.242 |
| | 2 | 3.249 | 3.21 | 3.253 | 3.227 | 3.688 | 3.412 | 3.202 | 3.267 | **3.198** | 3.457 | 3.295 | 3.33 | 3.308 | 3.3 |
| | 3 | 3.223 | 3.182 | 3.235 | 3.202 | 3.696 | 3.406 | 3.189 | 3.251 | **3.179** | 3.466 | 3.289 | 3.315 | 3.29 | 3.284 |
| | 4 | 3.135 | 3.097 | 3.145 | 3.119 | 3.58 | 3.311 | 3.1 | 3.165 | **3.091** | 3.366 | 3.185 | 3.222 | 3.196 | 3.195 |
| | 5 | 3.431 | 3.399 | 3.434 | 3.406 | 3.895 | 3.604 | 3.386 | 3.446 | **3.383** | 3.624 | 3.497 | 3.506 | 3.49 | 3.48 |
| | 6 | 3.067 | **3.025** | 3.078 | 3.051 | 3.522 | 3.248 | 3.036 | 3.099 | 3.026 | 3.307 | 3.118 | 3.158 | 3.136 | 3.133 |
| | 7 | 3.225 | 3.185 | 3.228 | 3.201 | 3.644 | 3.385 | 3.182 | 3.252 | **3.178** | 3.436 | 3.267 | 3.312 | 3.289 | 3.288 |
| | 8 | 3.356 | 3.321 | 3.358 | 3.33 | 3.793 | 3.513 | 3.305 | 3.379 | **3.304** | 3.552 | 3.401 | 3.436 | 3.419 | 3.408 |
| | 9 | 3.27 | 3.232 | 3.27 | 3.242 | 3.691 | 3.426 | **3.222** | 3.3 | 3.222 | 3.47 | 3.308 | 3.357 | 3.336 | 3.33 |
| | 10 | 3.269 | 3.228 | 3.273 | 3.246 | 3.708 | 3.431 | 3.226 | 3.29 | **3.219** | 3.476 | 3.316 | 3.347 | 3.327 | 3.32 |
| | 11 | 3.136 | **3.093** | 3.141 | 3.12 | 3.575 | 3.312 | 3.111 | 3.162 | 3.095 | 3.357 | 3.193 | 3.218 | 3.204 | 3.198 |
| | 12 | 3.124 | 3.081 | 3.13 | 3.1 | 3.538 | 3.275 | **3.069** | 3.152 | 3.076 | 3.342 | 3.151 | 3.215 | 3.191 | 3.191 |
| | Avg | 3.222 | 3.183 | 3.228 | 3.201 | 3.666 | 3.391 | 3.182 | 3.247 | **3.176** | 3.439 | 3.273 | 3.307 | 3.286 | 3.281 |
| Fold 3 | 1 | 3.171 | 3.152 | 3.167 | 3.162 | 3.64 | 3.374 | 3.199 | **3.149** | 3.161 | 3.427 | 3.288 | 3.199 | 3.249 | 3.208 |
| | 2 | 3.237 | 3.219 | 3.232 | 3.221 | 3.668 | 3.413 | 3.249 | **3.212** | 3.224 | 3.472 | 3.338 | 3.247 | 3.308 | 3.265 |
| | 3 | 3.212 | 3.193 | 3.21 | 3.2 | 3.671 | 3.407 | 3.237 | **3.193** | 3.207 | 3.474 | 3.334 | 3.241 | 3.296 | 3.254 |
| | 4 | 3.128 | 3.108 | 3.125 | 3.114 | 3.557 | 3.307 | 3.144 | **3.105** | 3.119 | 3.373 | 3.231 | 3.146 | 3.197 | 3.165 |
| | 5 | 3.419 | 3.4 | 3.409 | 3.403 | 3.878 | 3.608 | 3.441 | **3.398** | 3.402 | 3.655 | 3.519 | 3.427 | 3.492 | 3.436 |
| | 6 | 3.058 | 3.043 | 3.056 | 3.048 | 3.5 | 3.246 | 3.079 | **3.031** | 3.056 | 3.311 | 3.165 | 3.082 | 3.134 | 3.103 |
| | 7 | 3.214 | **3.194** | 3.206 | 3.196 | 3.618 | 3.385 | 3.23 | 3.2 | 3.209 | 3.452 | 3.317 | 3.233 | 3.292 | 3.26 |
| | 8 | 3.344 | **3.322** | 3.334 | 3.324 | 3.772 | 3.517 | 3.355 | 3.324 | 3.332 | 3.573 | 3.443 | 3.351 | 3.415 | 3.372 |
| | 9 | 3.256 | 3.235 | 3.245 | **3.234** | 3.667 | 3.427 | 3.27 | 3.241 | 3.253 | 3.489 | 3.357 | 3.269 | 3.334 | 3.297 |
| | 10 | 3.256 | 3.236 | 3.25 | 3.241 | 3.688 | 3.431 | 3.271 | **3.236** | 3.246 | 3.495 | 3.358 | 3.271 | 3.327 | 3.292 |
| | 11 | 3.122 | 3.111 | 3.123 | 3.108 | 3.551 | 3.307 | 3.15 | **3.107** | 3.122 | 3.36 | 3.226 | 3.156 | 3.203 | 3.168 |
| | 12 | 3.111 | 3.09 | 3.109 | 3.096 | 3.509 | 3.274 | 3.114 | **3.083** | 3.109 | 3.35 | 3.211 | 3.123 | 3.203 | 3.154 |
| | Avg | 3.211 | 3.192 | 3.206 | 3.196 | 3.643 | 3.391 | 3.228 | **3.19** | 3.204 | 3.453 | 3.316 | 3.229 | 3.287 | 3.248 |
| Fold 4 | 1 | 3.179 | **3.137** | 3.15 | 3.158 | 3.63 | 3.375 | 3.18 | 3.164 | 3.152 | 3.347 | 3.216 | 3.211 | 3.236 | 3.175 |
| | 2 | 3.247 | **3.208** | 3.216 | 3.226 | 3.654 | 3.414 | 3.233 | 3.221 | 3.214 | 3.388 | 3.265 | 3.265 | 3.29 | 3.232 |
| | 3 | 3.219 | **3.179** | 3.19 | 3.2 | 3.662 | 3.412 | 3.221 | 3.201 | 3.197 | 3.388 | 3.256 | 3.254 | 3.28 | 3.226 |
| | 4 | 3.142 | **3.1** | 3.11 | 3.121 | 3.549 | 3.313 | 3.132 | 3.121 | 3.109 | 3.296 | 3.164 | 3.167 | 3.189 | 3.134 |
| | 5 | 3.422 | **3.386** | 3.389 | 3.399 | 3.865 | 3.602 | 3.413 | 3.4 | 3.398 | 3.568 | 3.439 | 3.428 | 3.461 | 3.408 |
| | 6 | 3.076 | **3.03** | 3.045 | 3.055 | 3.491 | 3.251 | 3.067 | 3.048 | 3.042 | 3.237 | 3.097 | 3.109 | 3.129 | 3.071 |
| | 7 | 3.228 | **3.187** | 3.191 | 3.205 | 3.61 | 3.391 | 3.22 | 3.209 | 3.201 | 3.37 | 3.248 | 3.255 | 3.285 | 3.225 |
| | 8 | 3.35 | **3.313** | 3.313 | 3.329 | 3.757 | 3.516 | 3.339 | 3.328 | 3.321 | 3.486 | 3.366 | 3.368 | 3.399 | 3.34 |
| | 9 | 3.266 | **3.227** | 3.229 | 3.243 | 3.652 | 3.431 | 3.256 | 3.251 | 3.242 | 3.402 | 3.285 | 3.293 | 3.322 | 3.259 |
| | 10 | 3.269 | **3.227** | 3.233 | 3.245 | 3.674 | 3.438 | 3.257 | 3.246 | 3.239 | 3.411 | 3.289 | 3.286 | 3.317 | 3.263 |
| | 11 | 3.143 | **3.101** | 3.116 | 3.119 | 3.55 | 3.319 | 3.137 | 3.127 | 3.115 | 3.297 | 3.171 | 3.167 | 3.199 | 3.137 |
| | 12 | 3.126 | **3.084** | 3.09 | 3.107 | 3.497 | 3.276 | 3.105 | 3.091 | 3.089 | 3.262 | 3.135 | 3.157 | 3.173 | 3.12 |
| | Avg | 3.222 | **3.182** | 3.189 | 3.2 | 3.632 | 3.395 | 3.213 | 3.2 | 3.193 | 3.371 | 3.244 | 3.246 | 3.273 | 3.216 |
| All Folds | Avg | 3.224 | 3.195 | 3.214 | 3.202 | 3.654 | 3.389 | 3.212 | 3.206 | **3.194** | 3.414 | 3.28 | 3.251 | 3.272 | 3.242 |

**Fig. B.2.** Compression performance in (bpp) for evaluating models trained on the **reduced** neighbourhood sequences applied in Section 4.4 on **Dataset2**. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 3.025 bpp (Green) to the lowest compression 3.915 bpp (Red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

## References


Anagnostakos, K., Knafo, Y., Houfani, F., Zaharia, B., Egrise, F., Clerc-Urmès, I., et al. (2019). Value of 3D preoperative planning for primary total hip arthroplasty based on biplanar weightbearing radiographs. *BioMed Research International*, *2019*, Article 1932191. http://dx.doi.org/10.1155/2019/1932191.

Bishop, C. M. (2007). *Information science and statistics*, *Pattern recognition and machine learning* (5th ed.). Springer, URL https://www.worldcat.org/oclc/71008143.

Bossen, F., Flynn, D., Sharman, K., & Sühring, K. (2019). HEVC format range extension (RExt). Fraunhofer Heinrich Hertz Institute. URL https://hevc.hhi.fraunhofer.de/rext. (Accessed: 15 July 2019).

Brownlee, J. (2017). *Long short-term memory networks with python: develop sequence prediction models with deep learning*. Jason Brownlee, URL https://books.google.com.sa/books?id=ONpdswEACAAJ.

Cardenas, C., Mohamed, A., Sharp, G., Gooding, M., Veeraraghavan, H., & Yang, J. (2019). The cancer imaging archive. In *Data from AAPM RT-MAC grand challenge*. http://dx.doi.org/10.7937/tcia.2019.bcfjqfqb,

Cardenas, C. E., Mohamed, A. S. R., Yang, J., Gooding, M., Veeraraghavan, H., Kalpathy-Cramer, J., et al. (2020). Head and neck cancer patient images for determining auto-segmentation accuracy in T2-weighted magnetic resonance imaging through expert manual segmentations. *Medical Physics*, *47*(5), 2317–2322. http://dx.doi.org/10.1002/mp.13942.

Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555. URL http://arxiv.org/abs/1412.3555. arXiv:1412.3555.

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., et al. (2013). The cancer imaging archive (TCIA): Maintaining and operating a public information repository. *Journal of Digital Imaging*, *26*(6), http://dx.doi.org/10.1007/s10278-013-9622-7.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/n19-1423.

Dixon, S. (2019). *Diagnostic imaging dataset statistical release*: *Technical report*, UK: NHS England, URL https://www.england.nhs.uk/statistics/statistical-work-areas/diagnostic-imaging-dataset/.

e. V. OFFIS (2020). *Dcmtk is a collection of libraries, which includes dcmcjpls tool for encoding dicom file to jpeg-ls transfer syntax*. 26121 Oldenburg, Germany: e.V. OFFIS, URL https://support.dcmtk.org/docs/dcmcjpls.html. (Accessed: 15 July 2020).

Flynn, D., Marpe, D., Naccari, M., Nguyen, T., Rosewarne, C., Sharman, K., et al. (2016). Overview of the range extensions for the HEVC standard: Tools, profiles, and performance. *IEEE Transactions on Circuits and Systems for Video Technology*, *26*(1), 4–19. http://dx.doi.org/10.1109/TCSVT.2015.2478707.

Goodfellow, I. J., Bengio, Y., & Courville, A. C. (2016). *Adaptive computation and machine learning, Deep learning*. MIT Press, URL http://www.deeplearningbook.org/.

Goyal, M., Tatwawadi, K., Chandak, S., & Ochoa, I. (2019). DeepZip: Lossless data compression using recurrent neural networks. In A. Bilgin, M. W. Marcellin, J. Serra-Sagristà, & J. A. Storer (Eds.), *Data compression conference* (p. 575). IEEE, http://dx.doi.org/10.1109/DCC.2019.00087.

Goyal, M., Tatwawadi, K., Chandak, S., & Ochoa, I. (2020). DZip: Improved general-purpose lossless compression based on novel neural network modeling. In A. Bilgin, M. W. Marcellin, J. Serra-Sagristà, & J. A. Storer (Eds.), *Data compression conference* (p. 372). IEEE, http://dx.doi.org/10.1109/DCC47342.2020.00065.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hu, Y., Yang, W., Ma, Z., & Liu, J. (2020). Learning end-to-end lossy image compression: A benchmark. CoRR, abs/2002.03711. URL https://arxiv.org/abs/2002.03711. arXiv:2002.03711.

Hussain, A. J., Al-Fayadh, A., & Radi, N. (2018). Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, *300*, 44–69. http://dx.doi.org/10.1016/j.neucom.2018.02.094.

Kingma, F. H., Abbeel, P., & Ho, J. (2019). Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In K. Chaudhuri, & R. Salakhutdinov (Eds.), *Proceedings of machine learning research*: *vol. 97*, *Proceedings of the 36th international conference on machine learning* (pp. 3408–3417). PMLR, URL http://proceedings.mlr.press/v97/kingma19a.html.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), *3rd International conference on learning representations, conference track proceedings*. URL http://arxiv.org/abs/1412.6980.

Knoll, B. (2020). lstm-compress: data compression using lstm. URL https://github.com/byronknoll/lstm-compress. (Accessed: 15 July 2020).

Koff, D. A., & Shulman, H. (2006). An overview of digital compression of medical images: can we use lossy image compression in radiology? *Canadian Association of Radiologists Journal*, *57*(4), 211.

Lai, W., Huang, J., Ahuja, N., & Yang, M. (2017). Deep Laplacian pyramid networks for fast and accurate super-resolution. In *2017 IEEE conference on computer vision and pattern recognition* (pp. 5835–5843). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2017.618.

Liu, D., Li, Y., Lin, J., Li, H., & Wu, F. (2020). Deep learning-based video coding: A review and a case study. *ACM Computing Surveys*, *53*(1), 11:1–11:35. http://dx.doi.org/10.1145/3368405.

Lucas, L. F. R., Rodrigues, N. M. M., da Silva Cruz, L. A., & de Faria, S. M. M. (2017). Lossless compression of medical images using 3-D predictors. *IEEE Transactions on Medical Imaging*, *36*(11), 2250–2260. http://dx.doi.org/10.1109/TMI.2017.2714640.

Magli, E., Olmo, G., & Quacchio, E. (2004). Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC. *IEEE Geoscience and Remote Sensing Letters*, *1*(1), 21–25. http://dx.doi.org/10.1109/LGRS.2003.822312.

Matsuda, I., Mori, H., & Itoh, S. (2000). Lossless coding of still images using minimum-rate predictors. In *Proceedings of the 2000 international conference on image processing* (pp. 132–135). IEEE, http://dx.doi.org/10.1109/ICIP.2000.900912.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In T. Kobayashi, K. Hirose, & S. Nakamura (Eds.), *INTERSPEECH 2010, 11th Annual conference of the international speech communication association* (pp. 1045–1048). ISCA, URL http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

Nagoor, O. H., Whittle, J., Deng, J., Mora, B., & Jones, M. W. (2020a). Lossless compression for volumetric medical images using deep neural network with local sampling. In *IEEE international conference on image processing* (pp. 2815–2819). IEEE, http://dx.doi.org/10.1109/ICIP40778.2020.9191031.

Nagoor, O. H., Whittle, J., Deng, J., Mora, B., & Jones, M. W. (2020b). MedZip: 3D medical images lossless compressor using recurrent neural network (LSTM). In *25th International conference on pattern recognition* (pp. 2874–2881). IEEE, http://dx.doi.org/10.1109/ICPR48806.2021.9413341.

van den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In M. Balcan, & K. Q. Weinberger (Eds.), *JMLR workshop and conference proceedings*: *vol. 48*, *Proceedings of the 33nd international conference on machine learning* (pp. 1747–1756). JMLR.org, URL http://proceedings.mlr.press/v48/oord16.html.

OpenJPEG (2019). OpenJPEG An open-source JPEG 2000 codec written in C. Université de Louvain (UCL). URL https://www.openjpeg.org/. (Accessed: 15 July 2019).

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., et al. (2018). Image transformer. In J. Dy, A. Krause (Eds.), *Proceedings of machine learning research*: *vol. 80*, *Proceedings of the 35th international conference on machine learning* (pp. 4055–4064). PMLR, URL https://proceedings.mlr.press/v80/parmar18a.html.

Patidar, G., Kumar, S., & Kumar, D. (2020). A review on medical image data compression techniques. In *2nd International conference on data, engineering and applications* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/IDEA49133.2020.9170679.

Pavlov, I. (2019). *7-Zip is a file archiver with a high compression ratio*. 7-Zip, URL https://www.7-zip.org/. (Accessed: 15 June 2019).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9.

Reed, S. E., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., et al. (2017). Parallel multiscale autoregressive density estimation. In D. Precup, Y. W. Teh (Eds.), *Proceedings of machine learning research*: *vol. 70*, *Proceedings of the 34th international conference on machine learning* (pp. 2912–2921). PMLR, URL http://proceedings.mlr.press/v70/reed17a.html.

Rhee, H., Jang, Y. I., Kim, S., & Cho, N. I. (2020). Channel-wise progressive learning for lossless image compression. In *IEEE international conference on image processing* (pp. 1113–1117). IEEE, http://dx.doi.org/10.1109/ICIP40778.2020.9191322.

Salimans, T., Karpathy, A., Chen, X., & Kingma, D. P. (2017). PixelCNN++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International conference on learning representations, conference track proceedings*. OpenReview.net, URL https://openreview.net/forum?id=BJrFC6ceg.

Santurkar, S., Budden, D. M., & Shavit, N. (2018). Generative compression. In *2018 Picture coding symposium* (pp. 258–262). IEEE, http://dx.doi.org/10.1109/PCS.2018.8456298.

Schelkens, P., Munteanu, A., Tzannes, A., & Brislawn, C. M. (2006). JPEG2000. part 10. Volumetric data encoding. In *International symposium on circuits and systems*. IEEE, http://dx.doi.org/10.1109/ISCAS.2006.1693474, 4 pp.–3877.

Schiopu, I., Huang, H., & Munteanu, A. (2020). CNN-based intra-prediction for lossless HEVC. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(7), 1816–1828. http://dx.doi.org/10.1109/TCSVT.2019.2940092.

Schiopu, I., & Munteanu, A. (2020a). Deep-learning-based lossless image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(7), 1829–1842. http://dx.doi.org/10.1109/TCSVT.2019.2909821.

Schiopu, I., & Munteanu, A. (2020b). A study of prediction methods based on machine learning techniques for lossless image coding. In *IEEE international conference on image processing* (pp. 3324–3328). IEEE, http://dx.doi.org/10.1109/ICIP40778.2020.9190696.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423. http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x.

Sridhar, V. (2014). A review of the effective techniques of compression in medical image processing. *International Journal Of Computer Applications*, *97*(6).

Sullivan, G. J., Ohm, J., Han, W., & Wiegand, T. (2012). Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, *22*(12), 1649–1668. http://dx.doi.org/10.1109/TCSVT.2012.2221191.

Taubman, D. S., & Marcellin, M. W. (2002). *The Kluwer international series in engineering and computer science*: *vol. 642*, *JPEG2000 - image compression fundamentals, standards and practice*. Kluwer, http://dx.doi.org/10.1007/978-1-4615-0799-4.

Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., et al. (2017). Full resolution image compression with recurrent neural networks. In *2017 IEEE conference on computer vision and pattern recognition* (pp. 5435–5443). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2017.577.

Townsend, J., Bird, T., & Barber, D. (2019). Practical lossless compression with latent variables using bits back coding. In *7th International conference on learning representations*. OpenReview.net, URL https://openreview.net/forum?id=ryE98iR5tm.

Weinberger, M. J., Seroussi, G., & Sapiro, G. (2000). The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, *9*(8), 1309–1324. http://dx.doi.org/10.1109/83.855427.

Wu, X., & Memon, N. D. (1996). CALIC-a context based adaptive lossless image codec. In *1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings* (pp. 1890–1893). IEEE Computer Society, http://dx.doi.org/10.1109/ICASSP.1996.544819.

Wu, X., & Memon, N. D. (2000). Context-based lossless interband compression-extending CALIC. *IEEE Transactions on Image Processing*, *9*(6), 994–1001. http://dx.doi.org/10.1109/83.846242.

Yi, X., Walia, E., & Babyn, P. S. (2019). Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, *58*, http://dx.doi.org/10.1016/j.media.2019.101552.