

3D Medical Image Lossless Compressor Using Deep Learning Approaches

Thesis by

Omniah Hussain Jameel Nagoor

Submitted to Swansea University In Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy



Swansea University
Prifysgol Abertawe

Swansea, United Kingdom

Department of Computer Science

(February 14, 2022)

EXAMINATION COMMITTEE PAGE

The thesis of Omniah Hussain Jameel Nagoor is approved by the examination committee

Committee Chairperson:

Prof. Monika Seisenberger
Department of Computer Science
Swansea University

Committee Members:

1. Prof. Abhir Bhalerao
Department of Computer Science
University of Warwick

2. Dr. Michael Edwards
Department of Computer Science
Swansea University

Supervisor:

Prof. Mark W. Jones
Department of Computer Science
Swansea University

DECLARATION

I declare that the work presented in this thesis has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

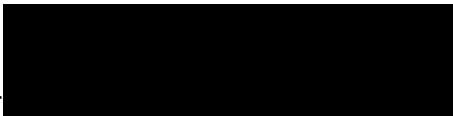
Swansea, February 14, 2022

A black rectangular box redacting the signature of Omniah Nagoor.

Omniah Nagoor

That this thesis is the result of my own investigations, except where otherwise stated and that other sources are acknowledged by footnotes giving explicit references and that a bibliography is appended.

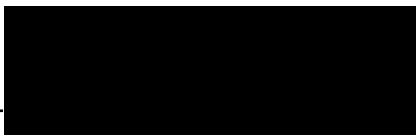
Swansea, February 14, 2022

A black rectangular box redacting the signature of Omniah Nagoor.

Omniah Nagoor

That I give consent for the thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Swansea, February 14, 2022

A black rectangular box redacting the signature of Omniah Nagoor.

Omniah Nagoor

©February, 2022

Omniah Hussain Jameel Nagoor

All Rights Reserved

ABSTRACT

3D Medical Image Lossless Compressor Using Deep Learning Approaches

Omniah Hussain Jameel Nagoor

The ever-increasing importance of accelerated information processing, communication, and storing are major requirements within the big-data era revolution. With the extensive rise in data availability, handy information acquisition, and growing data rate, a critical challenge emerges in efficient handling. Even with advanced technical hardware developments and multiple Graphics Processing Units (GPUs) availability, this demand is still highly promoted to utilise these technologies effectively. Health-care systems are one of the domains yielding explosive data growth. Especially when considering their modern scanners abilities, which annually produce higher-resolution and more densely sampled medical images, with increasing requirements for massive storage capacity. The bottleneck in data transmission and storage would essentially be handled with an effective compression method. Since medical information is critical and imposes an influential role in diagnosis accuracy, it is strongly encouraged to guarantee exact reconstruction with no loss in quality, which is the main objective of any lossless compression algorithm. Given the revolutionary impact of Deep Learning (DL) methods in solving many tasks while achieving the state of the art results, including data compression, this opens tremendous opportunities for contributions. While considerable efforts have been made to address lossy performance using learning-based approaches, less attention was paid to address lossless compression. This PhD thesis investigates and proposes novel learning-based approaches for compressing 3D medical images losslessly.

Firstly, we formulate the lossless compression task as a supervised sequential prediction problem, whereby a model learns a projection function to predict a target voxel given sequence of samples from its spatially surrounding voxels. Using such 3D local sampling information efficiently exploits spatial similarities and redundancies in a volumetric medical context by utilising such a prediction paradigm. The proposed NN-based data predictor is trained to minimise the differences with the original data values while the residual errors are encoded using arithmetic coding to allow lossless

reconstruction.

Following this, we explore the effectiveness of Recurrent Neural Networks (RNNs) as a 3D predictor for learning the mapping function from the spatial medical domain (16 bit-depths). We analyse Long Short-Term Memory (LSTM) models’ generalisability and robustness in capturing the 3D spatial dependencies of a voxel’s neighbourhood while utilising samples taken from various scanning settings. We evaluate our proposed MedZip models in compressing unseen Computerized Tomography (CT) and Magnetic Resonance Imaging (MRI) modalities losslessly, compared to other state-of-the-art lossless compression standards.

This work investigates input configurations and sampling schemes for a many-to-one sequence prediction model, specifically for compressing 3D medical images (16 bit-depths) losslessly. The main objective is to determine the optimal practice for enabling the proposed LSTM model to achieve a high compression ratio and fast encoding-decoding performance. A solution for a non-deterministic environments problem was also proposed, allowing models to run in parallel form without much compression performance drop. Compared to well-known lossless codecs, experimental evaluations were carried out on datasets acquired by different hospitals, representing different body segments, and have distinct scanning modalities (i.e. CT and MRI).

To conclude, we present a novel data-driven sampling scheme utilising weighted gradient scores for training LSTM prediction-based models. The objective is to determine whether some training samples are significantly more informative than others, specifically in medical domains where samples are available on a scale of billions. The effectiveness of models trained on the presented importance sampling scheme was evaluated compared to alternative strategies such as uniform, Gaussian, and sliced-based sampling.

ACKNOWLEDGEMENTS

“And say: My Lord! Increase me in knowledge.”

Al Quran – Surah Taha {20:114}

First and foremost, I praise my God, the Almighty, to grant me the opportunity to successfully accomplish this thesis work. Then, I would like to sincerely thank several people who helped, believed, and supported me in my thesis journey. Starting with my supervisor Prof. Mark Jones who guided me through seeking the knowledge of this degree. I would like to thank him for believing in me and providing me with his valuable comments, remarks and engagement through the learning process of my PhD. I also would like to express my most profound appreciation and gratitude to Dr Joss Whittle for being an exceptional supervisor. I would like to thank him for all the valuable advice and support during the difficult times. I am also thankful to Dr Jingjing Deng for his tips, advice, and help over the past few years.

My sincere thanks also go to my friends at Swansea University. Starting by an exceptionally close friend of mine, Hanen Emam, for being such a supportive friend. I would like to thank her for being there for me whenever I need her for helping, supporting and encouraging me during difficult times. My thanks also include my dear friends Mshael Alsheek, Arwa Aldagreeri, and Monirah Melhem, who always encourage and support me during my thesis work. I would like to offer my deepest thanks to my colleagues in the Computer Science department at Swansea University. It has indeed been a privilege to work with all of you.

I would like to include a unique and unlimited thanks to my lovely parents **Amal Al-Hindi** and **Hussain Nagoor**, my sisters, **Faigah**, **Shahad**, **Afnan**, and **Mawadah**, my brothers, **Mohammed** and **Ali**. I would like to thank them for pushing me toward my dreams, supporting me, motivating me and loving me. My gratitude and deep love go to my husband **Abdulrahman Merdad** and my son **Jawad Merdad** for supporting, believing in me and being patient during difficult times.

TABLE OF CONTENTS

Examination Committee Page	2
Declaration	3
Copyright	4
Abstract	5
Acknowledgements	7
List of Abbreviations	14
List of Figures	16
List of Tables	23
1 Introduction	25
1.1 Motivations	25
1.2 Thesis Statement	28
1.3 Objectives and Contributions	29
1.4 Thesis Structure	33
1.4.1 Chapter 1	33
1.4.2 Chapter 2	33
1.4.3 Chapter 4	33
1.4.4 Chapter 5	33
1.4.5 Chapter 6	33
1.4.6 Chapter 7	34
1.4.7 Chapter 8	34

2	Related Work	35
2.1	Overview	38
2.2	3D Medical Imaging	38
2.2.1	Representation and modalities	38
2.2.2	Analysis and Preprocessing	41
2.2.3	Visualization	43
2.2.3.1	Volume Rendering	44
2.2.3.2	Transfer Function	46
2.2.4	Challenges and Observations	49
2.3	Data Compression	50
2.3.1	Compression Basics	50
2.3.1.1	Lossy Data Compression	53
2.3.1.2	Lossless Data Compression	53
2.3.2	Compression Algorithms	54
2.3.2.1	Shannon Entropy	54
2.3.2.2	Prediction-Based Coding	55
2.3.2.2.1	Prediction by Partial Matching Coder	55
2.3.2.3	Dictionary-Based Coding	56
2.3.2.3.1	Run-Length Coding	56
2.3.2.3.2	Lempel-Ziv Algorithm	56
2.3.2.4	Statistical-Based Coding	58
2.3.2.4.1	Huffman coding	58
2.3.2.4.2	Arithmetic Coder	60
2.3.2.5	Transform-Based Coding	62
2.3.2.5.1	Discrete Fourier Transform (DFT)	63
2.3.2.5.2	Discrete Wavelet Transform (DWT)	64
2.3.2.5.3	Discrete Cosine Transform (DCT)	64
2.3.2.6	Quantisation	65
2.3.3	Image Compression	67
2.3.4	Compression Performance Metrics	67
2.3.4.1	Medical Image Compression	68
2.4	Deep Learning	69
2.4.1	Neural Networks	70
2.4.1.1	Vectorisation	71
2.4.1.2	Layers	72
2.4.1.3	Activation Functions	74

2.4.1.4	Loss Functions	77
2.4.1.5	Training by Back-Propagation	79
2.4.1.6	Optimizer	80
2.4.2	Recurrent Neural Networks (RNN)	83
2.4.2.1	Long-Short Term Memory (LSTM)	84
2.4.2.2	Gated Recurrent Unit (GRU)	85
2.4.3	Supervised Machine Learning	87
2.4.3.1	Sequence Prediction Models	87
2.4.4	Challenges and Observations	89
2.4.5	State-of-the-Art Image Compression Techniques	90
2.4.5.1	End-to-End Compression Frameworks (Lossy)	91
2.4.5.2	Entropy-based Compression Methods	91
2.4.5.3	Super-Resolution (SR) Compression Methods	94
2.4.5.4	Transform-based Compression Methods	95
2.4.6	Prediction-based Compression Methods	98
2.4.6.1	Classical Prediction-Based Methods	101
2.4.6.2	Learning-Based Prediction Methods	108
2.4.7	Challenges and Observations	111

3 Overview of Materials, Losses and Metrics **112**

3.1	Dataset Details	114
3.1.1	Dataset 1 - Private CT	115
3.1.2	Dataset 2 - Public CT	118
3.1.3	Dataset 3 - Public MRI	119
3.2	Loss Function	121
3.3	Evaluation Metrics	121
3.3.1	Compression Ratio	121
3.3.2	Compression Time	121

4 Lossless Compression for Volumetric Medical Images Using Deep Neural Network with Local Sampling **122**

4.1	Introduction	124
4.2	Proposed Method	126
4.2.1	Local Sampling	127
4.2.2	Transforming Function	129
4.3	Results and Discussion	129

4.3.1	Dataset Details	129
4.3.2	Comparisons with the state-of-the-art	130
4.4	Conclusion and Further Work	134
5	MedZip: 3D Medical Images Lossless Compressor Using Recurrent Neural Network (LSTM)	136
5.1	Introduction	138
5.2	Proposed Method	140
5.2.1	Overview	140
5.2.2	Network Architecture	141
5.2.3	Local Sampling	141
5.3	Results and Discussion	142
5.3.1	Dataset	142
5.3.2	The Proposed Models	142
5.3.3	Comparisons with the state-of-the-art	145
5.4	Conclusion and Further Work	150
6	Receptive Field Sampling for Learning-based 3D Medical Image Compression	152
6.1	Introduction	154
6.1.1	Contributions	155
6.2	Related Work	155
6.2.1	Classical Prediction Based Methods	156
6.2.2	Learning Based Prediction Methods	157
6.3	Proposed Method	160
6.3.1	Problem Description	160
6.3.2	Causal Neighbouring Sequence	160
6.3.3	Model	162
6.3.3.1	Training Hyper Parameters	162
6.3.4	Benchmarks	163
6.3.4.1	Benchmark1: Optimal Input Sequence (shape and size)	163
6.3.4.2	Benchmark2: Optimising Decoder Performance . . .	164
6.3.4.3	Benchmark3: Encoding-Decoding Performance . . .	165
6.3.4.4	Benchmark4: Sampling Strategy	165
6.3.4.5	Benchmark5: Compression Improvement During Training	165

6.3.4.6	Benchmark6: Comparing with State-of-The-Art Lossless Compression Methods	166
6.4	Results and Discussion	166
6.4.1	Dataset	166
6.4.2	Performance Metrics	168
6.4.3	Benchmark1: Optimal Input Sequence (shape and size)	168
6.4.4	Benchmark2: Optimising Decoder Performance	172
6.4.5	Benchmark3: Encoding-Decoding Performance	175
6.4.6	Benchmark4: Sampling Strategy	177
6.4.7	Benchmark5: Compression Improvement During Training	178
6.4.8	Benchmark6: Comparing with State-of-The-Art Lossless Compression Methods	179
6.4.9	Residual Plots	181
6.5	Conclusion and Further Work	186
7	Importance Sampling Based on Data-driven Information	187
7.1	Introduction	189
7.2	Related Work	190
7.2.1	Contributions in Enhancing SGD Optimiser & Sampling Selection	191
7.2.2	Contributions in Sampling Quality & Scoring Metrics	192
7.3	Methodology	194
7.3.1	Problem Description	194
7.3.2	Alternative Sampling Strategies	195
7.3.2.1	Random Sampling	195
7.3.2.2	Gaussian Sampling	195
7.3.2.3	Slice-based Sampling	197
7.3.3	Proposed Sampling Method	197
7.3.3.1	Background	197
7.3.3.2	Gradient-based Sampling	199
7.4	Experimental Results and Discussion	199
7.4.1	Performance Metrics	199
7.4.2	Dataset Details	199
7.4.3	Model Details	203
7.4.4	Gradient-based Sampling	203
7.4.4.1	Gradient-based Sampling Applied to Dataset1	203
7.4.4.2	Gradient-based Sampling Applied to Dataset2	212

7.4.5	Comparing with State-of-The-Art Lossless Compression Methods	215
7.5	Conclusion and Further Work	217
8	Concluding Remarks	218
8.1	Summary of Contributions	218
8.2	Future Research Work	222
	References	224
	Appendix A	244
	Appendix B	245
B.1	Chapter 4 - Additional Experimental Figures	245
B.2	Chapter 6 - Additional Experimental Figures	246

List of Abbreviations

2D 2-Dimensional.

3D 3-Dimensional.

4D 4-Dimensional.

Adam Adaptive Moment Estimation.

AI Artificial Intelligence.

bpp bits-per-pixel.

CALIC Context Based Adaptive Lossless Image Codec.

CNN Convolutional Neural Network.

CR Compression Ratio.

CT Computed Tomography.

DCT Discrete Cosine Transform.

DFT Discrete Fourier Transform.

DICOM Digital Imaging and Communications in Medicine.

DL Deep Learning.

DNN Deep Neural Network.

DWT Discrete Wavelet Transform.

FMRI Functional Magnetic Resonance Imaging.

GPUs Graphics Processing Units.

GRU Gated Recurrent Unit.

HEVC High Efficiency Video Coding.

HU Hounsfield Units.

JPEG-LS Joint Photographic Experts Group-Lossless.

JPEG2000 Joint Photographic Experts Group 2000.

LOCO-I LOw COmplexity LOssless COmpression for Images.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

ML Machine Learning.

MRI Magnetic Resonance Imaging.

MRP Minimum Rate Predictors.

MS-SSIM Multi Scale Structural Similarity.

MSE Mean Squared Error.

NLP Natural Language Processing.

NN Neural Network.

PET Positron Emission Tomography.

PSNR Peak Signal-to-Noise Ratio.

RLE Run-Length Encoding.

RNN Recurrent Neural Network.

ROI Region of Interest.

SGD Stochastic Gradient Descent.

LIST OF FIGURES

1.1	A summary overview demonstrates the growing pace of medical scans acquisition for the last decade (from 2013 to 2021) in the UK alone, according to the Diagnostic Imaging Dataset Statistical Release reports published by NHS.	26
2.1	An overview of the different between 2D scan and a 3D volume (series of 2D cross-sectional images).	38
2.2	An example of a modern CT scanner 2.2(a) with an overview of the scanning procedure 2.2(b).	39
2.3	An illustration of the complete pipeline for processing and reconstructing 3D CT data.	41
2.4	Illustrations of volume rendering (VR) of 3D CT volume and multiplanar view of three thin slices in the axial x-y plane (middle bottom), sagittal y-z plane (middle top), and coronal x-z plane (top right). The visualisation includes an example of volume element (voxel) and picture element (pixel).	43
2.5	Conceptual principle of volume visualization [1].	44
2.6	Examples of volume rendering of three different 3D medical image datasets.	45
2.7	A visual overview demonstrates the difference between a volume element (voxel) and a picture element (pixel).	45
2.8	A demonstration of rendering the same dataset using different transfer functions, whereas each mapping visually classifies different features of interest.	46
2.9	The main interface of DataPainter	47
2.10	A general block diagram of a data compression framework.	50
2.11	A general classification of data compression techniques.	51
2.12	Overview of LZ77 lossless compression algorithm [2].	57

2.13	An example of constructing a Huffman Tree for the symbols list $S = \{a, b, c, e\}$	59
2.14	Example of using Arithmetic Coding encoding and decoding process.	61
2.15	A general block diagram of a transform-based coding framework.	62
2.16	An example of a basic Multi-Layer Perceptron that composes of three layers types: input, hidden, and output, respectively.	70
2.17	An illustration of some well-known neural network layers' types, including dense layer, pooling layer, recurrent layer, convolutional layer, and deconvolutional layer.	72
2.18	An illustration of a gradient-based optimisation algorithm that minimises an objective function C by moving in the opposite direction (downhill) of weight's gradient $\frac{\partial C}{\partial W}$	80
2.19	Demonstration of a standard Recurrent Neural Network (RNN), which consists of multiple input observations (X_t), memory/internal states (h_t), and multiple output values (O_t).	83
2.20	An illustration of the different RNN cell classes, including traditional RNN unit 2.20(a), GRU unit 2.20(b), and LSTM unit 2.20(c).	86
2.21	An illustration of the numerous learning-based sequence prediction models classified by the input-output sequences' length.	87
2.22	Highlighting some current lossless codecs literature, including classical (non-learned) and deep learning-based methods.	90
2.23	Combined model architecture consisting of bootstrap and supporter models. Dense layers correspond to fully connected layers with ReLU activation. Linear layers are also fully connected layers but do not incorporate a non linear transformation. Concat block denotes concatenation of all the input vectors [3].	93
2.24	Model Overview. We propose lossless image compression through super resolution (SR). Our method first encodes a low-resolution image efficiently, and then leverages SR models to efficiently entropy-code the high-resolution images [4].	95
2.25	(a) Block diagram of the JPEG 2000 encoder algorithm. (b) Dataflow. [5]	95
2.26	Dyadic decomposition of a single tile [6].	97
2.27	Code-block partitioning of a wavelet volume (tile). The dark cube in the right volume represents a code-block [7].	97

2.28	A general block diagram for a lossless predictive-based compression framework with a detailed overview of the encoder and the decoder blocks.	99
2.29	A general illustration of the spatial information and predictive patterns that a prediction-based compression extracts and manipulates when compressing image/volume data. A classification of the numerous options utilised in the compression literature is proposed.	100
2.30	Block diagram for the JPEG-LS encoder [8].	101
2.31	An illustration of utilised local neighbouring causal pixels in JPEG-LS predictor (a) and CALIC predictor (b).	103
2.32	Typical HEVC video encoder (with decoder modeling elements shaded in light gray) [9]	105
2.33	3D-shaped support used for linear prediction in the proposed 3D-MRP algorithm. Pixels located at previous frames are indexed by the Z coordinate, with $Z = 0$ corresponding to the current frame. [10] . . .	106
2.34	Octree-based partitioning of a 3D-block, with one of the subblocks (in the bottom-right corner) being further divided into independent 2D-blocks, as proposed by the hybrid classification step of 3D-MRP algorithm. [10]	107
3.1	Overview of the voxel intensity histogram across all volumes (16 bit-depths) belonging to dataset 1.	115
3.2	Visualisations of three orthogonal slice views of some (16 bit-depths) sample volumes belong to dataset 1.	116
3.3	Overview of the histogram of voxel intensity in dataset 2 (16 bit-depths).118	
3.4	Visualisations of three orthogonal slice views of some (16 bit-depths) sample volumes belong to dataset 3.	120
4.1	An overview of the proposed lossless compression method.	126
4.2	Two neighbourhoods used for prediction. $z=0$ represents the current slice, the target voxel for prediction is black, grey voxels are used in the input sequence.	127
4.3	Overview of the histogram of voxel intensity across all volumes (16 bit-depths) belonging to the test set 1 (Orange) and test set 2 (Blue). 130	
4.4	An illustration of models loss function plots (i.e. training loss and testing loss) over epochs.	131

4.5	Comparing the compression ratio in bpp for the proposed models with the state-of-the-art lossless compression methods over 16-bits volumes on test set 1 (see table 4.3).	133
5.1	An overview of our proposed lossless compression framework using LSTM model.	138
5.2	Two different neighborhood shapes: a) 3D cube neighboring sequence and b) 3D pyramid neighboring sequence. $z=0$ represents the current slice.	143
5.3	The bits-per-pixel (bpp) for each lossless compression method was measured over TestSet1. Cells are highlighted from the maximum compression of 3.837 bpp (Blue) to the minimum compression of 6.236 bpp (Red).	146
5.4	An illustration of the bpp average and the standard deviation over TestSet1 groups for all the lossless compression methods from Fig. 5.3.	147
5.5	Illustrating the compression ratio in bpp for the proposed models compared to the state-of-the-art lossless compression methods on TestSet2 (16-bits volumes). Cells are highlighted from the maximum compression 2.949 bpp (Blue) to the minimum compression of 4.52 bpp (Red).	148
5.6	A summary overview of the compression performance over the two test sets for all the lossless methods.	150
6.1	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).	154
6.2	Overview of the optimized 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.	160
6.3	A demonstration of the proposed causal neighbouring sequences that can have different shapes, dimensions, block sizes, and sequence lengths.	161
6.4	A many-to-one standard LSTM model with a detailed overview of the LSTM unit. Each LSTM cell contains three main gates (input i_t , forget f_t , and output gate y_t), and a cell state C_t .	162
6.5	The order in which the decoder will decode voxels based on the defined causal neighbourhood specifications (i.e. sequentially, parallel batches, and diagonally).	164

6.6	A visualisation of some (16-bits) sample volumes from the two datasets used in this chapter, namely, Dataset1 and Dataset2 is shown. . . .	167
6.7	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 Benchmark 1 section 6.4.3.	169
6.8	Compression performance in (bpp) for evaluating the trained models with different neighbouring sequences from Benchmark 1 section 6.4.3 validated over unseen set Dataset2	170
6.9	Benchmark 2 result that illustrates the compression ratio in (bpp) for compressing Dataset1 using models trained on reduced neighbourhood sequences applied in section 6.4.4.	172
6.10	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.	173
6.11	Compression performance in (bpp) for evaluating models trained on the reduced neighbourhood sequences applied in section 6.4.4 on unseen set Dataset2	174
6.12	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	177
6.13	Benchmark5 result, which illustrates (bpp) variations during models’ training steps after 10, 20, 30, 40, 50 and 60 epochs.	178
6.14	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).	179
6.15	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 4.52 bpp (red).	180
6.16	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).	181

6.17	Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 1 while focusing on a specific region within the slice to highlight the impact.	183
6.18	Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 7 while focusing on a specific region within the slice to highlight the impact.	184
6.19	Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 10 while focusing on a specific region within the slice to highlight the impact.	185
7.1	The sampling acquisition strategies applied in this chapter, including <i>a)</i> Slice-based Sampling, <i>b)</i> Random (Uniform) Sampling, <i>c)</i> Gradient-based Sampling, and <i>d)</i> Gaussian Sampling.	196
7.2	Visualisations of some (16 bit-depths) sample volumes belong to Dataset1 . Each subfigure illustrates 3D volume visualisation of the patient’s entire Torso with three orthogonal slice views for colour mapping of slices’ intensity values (top row) and gradient magnitude values (bottom row).	201
7.3	Visualisations of some (16 bit-depths) sample volumes belong to Dataset2 . Each subfigure illustrates a patient’s head and neck with three orthogonal slice views for colour mapping of slices’ intensity values (top row) and gradient magnitude values (bottom row).	202
7.4	Gradient magnitudes histogram plots (linear & logarithmic scales) across Dataset1	205
7.5	Visualisations of the proposed gradient sampling schemes. Each subfigure illustrates a subsampling approach utilised in drawing training samples.	207
7.6	Bits-per-pixel (bpp) for compressing Dataset1 using models trained on different training subsets.	208
7.7	Compression performance in (bpp) for evaluating models’ trained with different sampling strategies validated over Dataset2	210
7.8	Gradient magnitudes histogram plots (linear & logarithmic scales across Dataset2	212
7.9	Bits-per-pixel (bpp) for compressing Dataset2 using models trained on different training subsets.	214

7.10	Illustrating the compression ratio in bpp of models trained with the some of proposed sampling schemes compared to the state-of-the-art lossless compression methods over Dataset1	215
7.11	Illustrating the compression ratio in bpp of all models trained with the proposed sampling schemes compared to the state-of-the-art lossless compression methods over Dataset2	216
B.1	Comparing the compression ratio in bpp for the proposed MLP models with the state-of-the-art lossless compression methods over 16-bits volumes on test set 1 ordered by the pixel spacing.	245
B.2	Another version of Benchmark 1 results compares the overall receptive density in 3D input options next to each other from Fig. 6.7 in section 6.4.3.	246
B.3	Another version of Benchmark 2 results compares the overall receptive density in 3D input options next to each other using models trained on reduced neighbourhood sequences from Fig. 6.9 in section 6.4.4. . .	247
B.4	An additional version of Fig. 6.10 reorders the sequence options to show the overlaps in the receptive field next to each other, which may give further insight into their contributions.	247
B.5	An illustration that compares model loss function plots (i.e. training loss and evaluation loss) with different pyramid input vectors (including left voxels from Benchmark1) over 60 epochs.	248
B.6	Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 1.	249
B.7	Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 7.	250
B.8	Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 10.	251

LIST OF TABLES

2.1	Some Hounsfield numbers for different human substance [11,12]. . . .	40
2.2	A general comparison between Lossy and Lossless compressing schemes.	52
2.3	A selection of activation functions commonly applied in neural networks.	75
2.4	The mode-selection options utilised in the JPEG-LS predictor [13] . .	102
2.5	A comprehensive exploration of the current literature for lossless predictive-based codecs is presented, including classical and learning-based approaches.	110
3.1	Overview of dataset 1 details composed of (16 bit-depths) CT medical images.	117
3.2	Overview of dataset 2 information composed of 16 bit-depth CT medical images [14,15].	118
3.3	Overview of dataset 3 composed of 12 MRI volumes of patients' head and neck scans (16 bit-depths) [16–18].	119
4.1	The proposed neural network architecture.	129
4.2	Illustrating the neural network training specifications for the two proposed models.	129
4.3	Compression performance in bpp for all datasets computed by our two proposed models and the existing state-of-the-art lossless methods for 16-bits volumes. The best compression result (bpp), is in bold.	134
5.1	The proposed LSTM architecture.	141
5.2	An overview of the training set and the training hyper parameters for each of the proposed LSTM models.	142
5.3	Comparing the compression performance (compression ratio (bpp) and compression time (s)) of different neighboring sequence (3D pyramid & 3D cube) with different block sizes.	143

6.1	The NN architecture used as the stander specification for all the proposed models.	162
6.2	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 from Benchmark1 and Benchmark2.	176
7.1	Dataset1 composed of 16 bit-depth medical images.	200
7.2	The NN architecture used as the stander specification for all the proposed models.	203
7.3	A summary overview of all the proposed gradient sampling schemes across Dataset1	204
7.4	A summary overview of all the proposed gradient sampling schemes across Dataset2	213
8.1	Overall summary table for the main contributions proposed in this PhD thesis.	221

Introduction

“The key to artificial intelligence has always been the representation.”

—Jeff Hawkins

1.1 Motivations

One of the biggest challenges in the big data and machine intelligence era is handling, understanding, storing and transmitting information efficiently. Such challenges led to vast fields of sciences to study and potentially expand on different levels. The increasing amount of data production can be effectively reduced by employing a coder algorithm. Reproducing information into more compact representations by utilising some knowledge or underlying structures known as data compression. This field of science is essential in many domain applications, including medical imaging. Given the large-scale of generated volumes, high scanning quality, and explosive annual growth of hospital data, the demand for developing efficient and robust algorithms rises. As these data are digitally available and archived, this creates numerous opportunities and requirements for designing intelligent algorithms not only for transmitting and storing but also for other tasks such as analysing, segmenting, classifying, or object detecting [19].

When examining the growing pace of medical scan acquisitions within the last decade, it is noticeable that the scanning productions are on the scale of millions in the UK alone for each year, as illustrated in Fig. 1.1. According to the Diagnostic



Figure 1.1: A summary overview demonstrates the growing pace of medical scans acquisition for the last decade (from 2013 to 2021) in the UK alone, according to the Diagnostic Imaging Dataset Statistical Release reports published by NHS. These statistical reports contain records across all types of medical imaging with various modalities, including Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Plain Radiography (X-Ray), and Ultrasound.

Imaging Dataset Statistical Release published by NHS, these statistical reports contain annual records across all types of imaging with various modalities, including Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Plain Radiography (X-Ray), and Ultrasound [20–28]. Such precise high-resolution medical images are vital for patient evaluation, diagnosis, and clinical applications. 3D advanced imaging has further been used to diagnose injuries, dysfunction or pathology, prognostic, and therapeutic decision-making for the medical and biomedical professions, including preoperative planning [29]. However, these images pose a substantial technical challenge from their ample storage requirements. Furthermore, due to data reliability and accuracy, it is not recommended to use lossy compression procedures for these domain-specific tasks, especially for clinical purposes. Since any loss in the original data may introduce artifacts, affect diagnoses performance, leading to misleading diagnoses, or unfavourable treatment [30, 31].

Most of the current state-of-the-art classical lossless codecs rely on hand-crafted or linear transformations, which have limited capacity in representing non-linear data distributions and correlations. On the other hand, deep learning techniques accomplish

remarkable gains in representing non-linearity and likelihood data estimation in many domain applications, including compression. Over the last decade, the use of machine learning methods has been widespread, given its impressive ability to learn and solve tasks within numerous areas. Examples of non-linear learned mapping functions include discovering underlying geometric structures, localised correlated features, complicated patterns, and inferencing implicit information. Leveraging the strength of learning-based approaches in lossless data reduction opens numerous opportunities in solving domain-specific applications but is not restricted to volumetric medical images, satellite images, and high-resolution videos [32–34].

The research development on compression utilising learning-based methods forms a promising research direction, especially when considering the growing volume of large-scale digitised images. Given its models’ ability to map non-linearity, estimate higher-dimensional data distributions, learn differentiable mapping functions, learning capacity, and generalisability, designing novel methods in this current trend forms great research potential. While a considerable effort has been made to address lossy performance using neural networks, less attention was paid to performing lossless compression. However, this area is gaining more popularity and contributions in the recent deep learning literature. Thus, the overarching motivation for this thesis aims to continue the current trend in exploring the production of novel learning-based methods, specifically for the domain of 3D medical imaging lossless reduction.

A large-scale model with deeply stacked layers is commonly utilised within the deep-learning domain, specifically in image analysis and features recognition breakthroughs tasks (e.g. image classifications or object detection). For instance, most state-of-the-art models (e.g. AlexNet [35], VGG16 [36], or ResNet101 [37]) are relatively large in models’ size (i.e. over 100MB). While these SOTA models produce consistently well-performing results, optimising all weight parameters during training would take time to tune, even with high-capability GPUs. Likewise, similar challenges arise when examining learning-based models utilised for data compression regardless of the sought compression quality (i.e. lossy or lossless). Such a process may take days or up to weeks to be completed, depending on how deep and large the model’s architecture is. Thus, smaller network architectures are often more desirable when considering the bottleneck of disk and bandwidth limitations. Therefore, we aim to propose learning-based solutions that are relatively small, making training faster to complete while still retaining a better compression performance by utilising localised structures and patterns within context.

One of the contributions that we seek in this thesis is selecting a representative

training subset by developing a new sampling methodology. The primary intent is to select a lower quantity of points while maintaining information for efficiently training a learning-based model. More than a billion candidate samples are available for training when formulating the compression task as a sequence prediction problem at a voxel scale. Subsampling to reduce training space forms a challenging yet highly demanded active area of research within the deep learning literature. Such development aims to optimise the training time by reducing computation cost and error rate while increasing convergence rate and training stability. Contributions to this research field can notably assist further numerous domains such as multi-agent systems [38], human-annotated labelling [39], and efficient subsampling of other domain-specific distributions [38, 40–42]. Thus, we aim to introduce a novel methodology for an offline data-driven subsampling scheme that increases the learned content and evaluation performance while reducing the training space.

1.2 Thesis Statement

“Deep Learning Can Make The Lossless Compression of 3D Medical Image Practically Efficient and Effective.”

The research questions that are answered to validate the thesis statement and achieve the objectives are:

1. Does involving learning-based strategies for coding medical volumes losslessly lead to higher bit-reduction?
Addressed in Chapter 4, Publication Outcome: [43].
2. Would the use of a compact learning-based sequence prediction model practically lead to better generalisability and higher compression performance?
Addressed in Chapter 5, Publication Outcome: [44].
3. To what extent can the learning-based sequence prediction codec be practically enhanced through the use of distinct local sampling grid options?
Addressed in Chapter 6, Publication Outcome: [45].
4. Is there a sampling selection scheme that draws critical training samples, and empirically causes a better learning-based codec performance?
Addressed in Chapter 7.

1.3 Objectives and Contributions

The current common process of reducing large-scale volumetric datasets is done by applying one of the state-of-the-art classical methods (e.g. JPEG-LS), which relies on handcrafted transformations with limited capacity in representing non-linearity and are generally practically inefficient. Compared to these limitations, facilitating learning-based approaches is more promising, given its remarkable breakthroughs in solving many domain problems. When leveraging a pre-trained learning model, generalisability is often gained across unseen datasets, forming one of its outstanding features.

To this end, this work’s overall objective is to develop novel learning-based compression frameworks that efficiently and practically reduce the storage requirements of volumetric medical images losslessly within healthcare systems. Such an encoding-decoding strategy would effectively represent the digitised volumes at lower bit rates without losing any essential information, allowing the hospital’s archiving system to gain more savings in storage. More benefits are related to bandwidth and communication services, allowing practical and rapid data transmission. The proposed mechanism exploits not only redundant information but also guarantees exact reconstruction without affecting the quality of the original volumes while preserving scans’ precision. In clinical applications, recovering the ground truth images from the compact representations would allow radiologists to examine scans precisely without influencing the overall diagnosis performance as no artifacts or noise would be introduced.

We seek to offer deep learning models that understand and exploit voxels’ spatial features and correlations in reducing the bit rate of those large-scale scans. We also comprehensively study the outcomes of aggregating various causal neighbouring localisation options (i.e. over different dimensions and coverages) on the model’s learned mapping function performance (i.e. compression ratio and time). The architectures of the learning-based proposed solutions are chosen to be relatively small in scale, optimising not only the bottleneck of disk and bandwidth limitations but also the model’s training time. Many enhancements to the framework’s procedure were gradually introduced to grant a rapid encoding-decoding mechanism while practically leveraging parallelisation. A further objective is a novel important sampling strategy, whereby downsizing sampling space is performed by utilising training samples’ quality while reducing quantity, resulting in fast convergence and retaining evaluation performance.

Furthermore, the proposed methods can possibly be applied to a wide range of other domain applications, including high-resolution data distributions such as im-

ages or videos. The usages can further extend to code other potential information sources that require preserving precision, such as forensic context [46], security and biometric authentication records [47, 48], remote sensing applications as monitoring forest fire [49], infrared thermal images [50], and high-resolution satellite images and videos [51].

The key contributions of this thesis folds in the following streams:

- Utilising deep learning techniques to introduce a compression framework specifically for the reduction of volumetric medical images (16 bit-depths).
- Introducing novel learning-based solutions, balancing architecture’s compactness and training time while still delivering favorable compression achievements.
- Demonstrating a thorough investigation on the use of various spatially localised features and correlations while highlighting impacts and influences on compression performance (i.e. compression time and compression ratio).
- Examine the generalisability of the prediction-based model when training on neighbourhood sampling sequences across multiple volumes with diverse scanning qualities, modalities and settings.
- Optimising encoding-decoding performance by presenting novel neighbouring input sequences and exploiting voxel spatial correlations while efficiently leveraging parallel implementation.
- An offline data-driven sampling scheme that utilises voxel gradient magnitude as a scoring metric is presented. The aim is to decrease the vast training space into a representative minimised set while still gaining a high compression ratio.
- Evaluating the compression performance of our proposed voxel-wise prediction models against state-of-the-art lossless compression methods across various information sources, scanner’s modalities, and patients’ bodies.

Outcomes from this thesis have also contributed to several publications as outlined in the List of Publications Appendix A. The key contributions of each paper related to the main body of work can be summarized as follows:

1. O. H. Nagoor, R. Borgo, and M. W. Jones, “Data Painter: A Tool for Colormap Interaction”, *Submitted to Computer Graphics & Visual Computing*, 2017, **Awarded The Best Full Paper Award**, [52].

We propose “Data Painter”: a novel colour mapping tool that allows interactive customization of nested colourmaps with a user-friendly interface specifically for high dynamic range datasets (e.g. thermal images). The primary intentions are to create more visually distinguishable representations and reveal hidden features while maximizing user perceptual reach and understanding of the underlying data. Our tool allows interactive placement and transformations of colourmaps in the data range with real-time visual feedback on the rendering window. Such procedure guides finding influential and representative colourmaps in less time while still highlighting more features and structural details. A new objective measurement was proposed to evaluate the generated dense colourmaps’ effectiveness compared to alternative standard colourmaps.

2. O. H. Nagoor, J. Whittle, J. Deng, B. Mora, M. W. Jones, “Lossless Compression for Volumetric Medical Images Using Deep Neural Network With Local Sampling”, *Submitted to The IEEE International Conference on Image Processing*, 2020, **Awarded ICIP 2020 Top Viewed Q&A Paper Award (2nd place)**, [43].

We introduce a novel learning-based codec that utilises the MLP model as a 3D predictor for the particular needs of reducing volumetric medical images (16-bits) losslessly. We train the NN-based model with two sequence types extracted from 3D local neighbourhood voxels (i.e. cubic-shaped and pyramid-shaped) while solving the supervised sequence prediction problem. Two models are trained to predict the next target voxel given several surrounding intensity values from its immediate local gride while adjusting models’ parameters. Such local sampling representations assist in learning context information and reducing spatial redundancies while minimising the differences with ground truth voxels. The resulted coding redundancies over the prediction errors will then be decreased using an arithmetic coder. We evaluate the compression performance of our proposed models to standard state-of-the-art lossless compression methods over two volumetric CT medical datasets.

3. O. H. Nagoor, J. Whittle, J. Deng, B. Mora and M. W. Jones, “MedZip: 3D Medical Images Lossless Compressor Using Recurrent Neural Network (LSTM),” *Submitted to The 25th International Conference on Pattern Recognition, 2021, [44]*.

We present another learning-based lossless codec (MedZip) that utilises the LSTM neural network to decorrelate 3D spatial information at the voxel level. The intention for choosing the LSTM cells is its ability to capture long dependencies within the 3D regions by using the gating mechanisms. Such gates allow LSTM to implicitly control data flow while flexibly memorising spatial features of 3D neighbouring voxels within the given sequences. We examine the selection of sequences across medical volumes with various scanning settings to assist the model’s generalisability while learning the input-output differentiable mapping functions. The generated residual errors are compressed to lower bit using an arithmetic coder. Comparison to state-of-the-art lossless compression standards, including well-known image and video codecs, was proposed to compress 3D volumetric medical images (16-bits). Empirical results reveal that MedZip demonstrates higher bit reduction and generalisability while evaluating across unseen modalities (i.e. CT and MRI).

4. O. H. Nagoor, J. Whittle, J. Deng, B. Mora and M. W. Jones, “Sampling Strategies for Learning-based 3D Medical Image Compression”, *Accepted for Elsevier Machine Learning with Applications Journal, 2022, [45]*.

We propose the first comprehensive study on voxel-wise prediction using input sampling schemes. We examine the impact of utilising numerous input configurations and sampling schemes on the many-to-one sequence prediction model, specifically for lossless medical imagery reduction (16-bit depths). The main objective is to determine the optimal practice for enabling the learning-based model to achieve a high compression ratio and fast encoding-decoding performance. By omitting the left voxels from input sequences, we allow the model to leverage parallelism while speeds-up decompression up to $37\times$. We evaluate our predictive codecs’ compression performance (i.e. bpp and time in seconds) compared to many state-of-the-art lossless alternatives, including learning-based and classical codecs. The experimental measurements were carried out on datasets acquired by different hospitals, representing different body segments and with distinct scanning modalities (i.e. CT and MRI).

1.4 Thesis Structure

1.4.1 Chapter 1

The remainder of chapter 1 summarizes the thesis’s structure with its primary contributions.

1.4.2 Chapter 2

Background [52]: Chapter 2 provides the necessary related work, fundamentals, and background information, including knowledge about 3D medical imaging, deep learning models, and data compression algorithms with a detailed overview of some state-of-the-art classical and learning-based compression approaches. Also, at the beginning of chapters 5 and 6 additional recent literature reviews pertaining specifically to each chapter are provided.

1.4.3 Chapter 4

NN Prediction-based Framework [43]: Chapter 4 introduces the initial prediction-based compression framework utilising a neural network model to learn a differentiable projection function and exploit the volume’s 3D localised features at the voxel level. This chapter also intended to overview the two unique shapes of the 3D neighbouring blocks utilised as input sequences fed to the proposed model during training. Experimental evaluation of the proposed system across various 3D datasets is given and discussed while comparing to alternative competing traditional lossless compressors.

1.4.4 Chapter 5

LSTM Prediction-based Framework (MedZip) [44]: This chapter presents another prediction-based compression framework that employs LSTM blocks to leverage its strength in memorising long dependencies within the given input sequences. Additionally, two local sampling approaches were empirically aggregated while examining their effectiveness in reflecting the 3D spatial correlations. The impact of using LSTM as a 3D predictor was estimated based on the learned input-output mapping function. Experimental results of the proposed codec are provided and compared to some state-of-the-art classical methods, including well-known image and video coders. The validation also includes comparing compression ratio and performance to a competing deep learning method.

1.4.5 Chapter 6

A Comprehensive Study on Sampling Strategies [45]: Chapter 6 introduces a comprehensive comparison to benchmark voxel-wise prediction models. The main intention is to determine the optimal practice for enabling the proposed LSTM model to achieve a high compression ratio and fast encoding-decoding performance. The study examines numerous causal neighbourhood options of the input sequence while

highlighting performance effects and trade-offs. Also, a novel sequence configuration was offered to reduce execution time while speeding up the decoder. Experimental results were carried out on datasets acquired by different hospitals, scanning modalities, and body segments. Moreover, validations opposed to some state-of-the-art methods are included.

1.4.6 Chapter 7

Gradient-based Importance Sampling: Given that the choice of training samples plays a crucial role in updating the model’s parameters and overall learned input-output mapping function, this chapter proposes a novel gradient-based sampling scheme to draw less quantity yet more informative training instances. The primary goal is to find a sufficient subset characterising the prediction-based model’s parameters to accomplish the best bpp reduction while reducing the computation cost. A variety of subsampling schemes was experimentally evaluated and discussed over various datasets to demonstrate efficiency compared to alternative well-known sampling methods.

1.4.7 Chapter 8

Conclusions and Future Work: A conclusion of the thesis’s chapters summarises all the proposed methods, remarks, and key findings. In addition, possible extensions and potential further developments are included.

Related Work



Contents

2.1	Overview	38
2.2	3D Medical Imaging	38
2.2.1	Representation and modalities	38
2.2.2	Analysis and Preprocessing	41
2.2.3	Visualization	43
2.2.3.1	Volume Rendering	44
2.2.3.2	Transfer Function	46
2.2.4	Challenges and Observations	49
2.3	Data Compression	50
2.3.1	Compression Basics	50
2.3.1.1	Lossy Data Compression	53
2.3.1.2	Lossless Data Compression	53
2.3.2	Compression Algorithms	54
2.3.2.1	Shannon Entropy	54
2.3.2.2	Prediction-Based Coding	55
2.3.2.2.1	Prediction by Partial Matching Coder	55
2.3.2.3	Dictionary-Based Coding	56
2.3.2.3.1	Run-Length Coding	56
2.3.2.3.2	Lempel-Ziv Algorithm	56
2.3.2.4	Statistical-Based Coding	58
2.3.2.4.1	Huffman coding	58
2.3.2.4.2	Arithmetic Coder	60
2.3.2.5	Transform-Based Coding	62
2.3.2.5.1	Discrete Fourier Transform (DFT)	63
2.3.2.5.2	Discrete Wavelet Transform (DWT)	64
2.3.2.5.3	Discrete Cosine Transform (DCT)	64
2.3.2.6	Quantisation	65
2.3.3	Image Compression	67
2.3.4	Compression Performance Metrics	67
2.3.4.1	Medical Image Compression	68
2.4	Deep Learning	69
2.4.1	Neural Networks	70
2.4.1.1	Vectorisation	71
2.4.1.2	Layers	72
2.4.1.3	Activation Functions	74
2.4.1.4	Loss Functions	77

2.4.1.5	Training by Back-Propagation	79
2.4.1.6	Optimizer	80
2.4.2	Recurrent Neural Networks (RNN)	83
2.4.2.1	Long-Short Term Memory (LSTM)	84
2.4.2.2	Gated Recurrent Unit (GRU)	85
2.4.3	Supervised Machine Learning	87
2.4.3.1	Sequence Prediction Models	87
2.4.4	Challenges and Observations	89
2.4.5	State-of-the-Art Image Compression Techniques	90
2.4.5.1	End-to-End Compression Frameworks (Lossy)	91
2.4.5.2	Entropy-based Compression Methods	91
2.4.5.3	Super-Resolution (SR) Compression Methods	94
2.4.5.4	Transform-based Compression Methods	95
2.4.6	Prediction-based Compression Methods	98
2.4.6.1	Classical Prediction-Based Methods	101
2.4.6.2	Learning-Based Prediction Methods	108
2.4.7	Challenges and Observations	111

2.1 Overview

This chapter will review the literature on 3D Medical Imaging, Deep Learning, and Data Compression that form the foundation of the topics discussed throughout the remainder of this thesis.

2.2 3D Medical Imaging

2.2.1 Representation and modalities

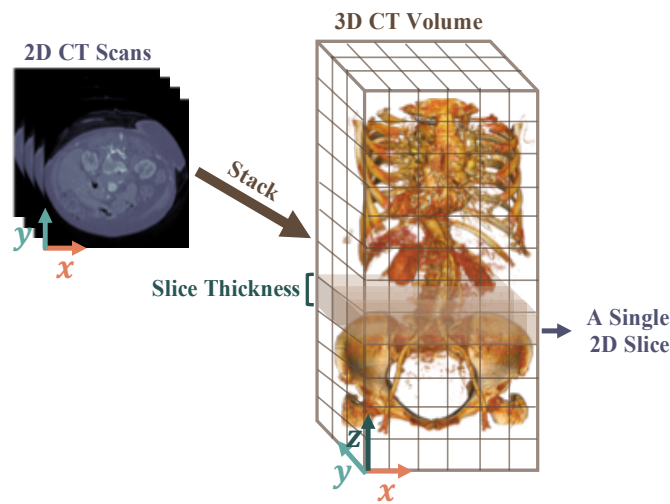


Figure 2.1: An overview of the different between 2D scan and a 3D volume (series of 2D cross-sectional images).

A brief overview of the medical image representation, modalities, processing, applications, and visualization is provided in this section. Medical images are a particular type of scanned photos that capture a patient’s entire body or region. Different types of modalities are produced based on the scanning technologies, revealing different aspects of the body’s internal structures, organs, injuries, functionalities, conditions or diseases. Essentially, these visual representations are commonly created for purposes such as diagnosis, treatment and presurgery planning. X-ray, Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Functional Magnetic Resonance Imaging (fMRI), Positron Emission Tomography (PET), and Ultrasound (US) are some examples of the numerous scanning physical technologies. Choosing which modality to apply is case dependent. The dimensionality of these images varies; for instance,

X-rays generate 2-Dimensional (2D) images while other modalities generate higher-dimensional representations 3-Dimensional (3D) images (e.g. CT, MRI, and PET), or even 4-Dimensional (4D) (e.g. Ultrasound). 3D imaging is considered as stack of 2D images or a series of 2D cross-sectional images (e.g. 3D CT scans are a stack of 2D X-rays images as shown in Fig 2.1). The main intention of creating the 3D volumetric medical images is its high flexibility in exploring, representing, and reformatting the Region of Interest (ROI) in various reconstruction planes. Generating a 2D image (projection image) is produced by passing a beam through a patient’s body from a fixed angle and superimposing all structures in its path onto a detector. In contrast, creating 3D axial cross-sectional slices, multiple beams are passed through the body, whereas each beam is from a different angle. In CT scanning, this process is known as digital geometry processing. An overview of the scanning procedure is illustrated in Fig. 2.2(b), whereas the X-ray source progressively rotates around the scanned object (e.g. human body) and projects cross-sectional images to the detectors. Also, an example of modern CT scanner is presented in the same Fig. 2.2(a). In the rest of this section, we will first overview the CT 3D volume representation, processing, and visualization as it is the primary dataset type applied in the thesis’s experiments. Also, a brief overview of the MRI modality will be given.

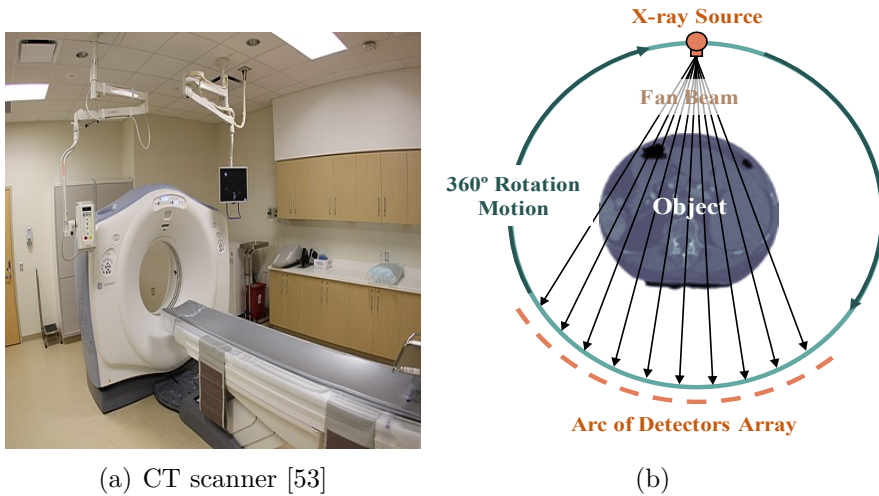


Figure 2.2: An example of a modern CT scanner 2.2(a) with an overview of the scanning procedure 2.2(b).

CT data is a well-known image modality with a great visualisation of bones, internal organs and blood vessels. The soft tissues, bones, water, fats, and air are defined as Hounsfield scales or CT numbers. Typically, this quantitative measure of radiodensity is described to evaluate CT scans, known as Hounsfield Units (HU) —

named after Sir Godfrey Newbold Hounsfield [54]. A linear density scale represents CT intensity values in the HU standardisation, where -1000 is air, 0 is water, and $+300$ and higher represent bones. Table 2.1 illustrates the ranges of some Hounsfield units used for some CT intensity values [11, 12]. While scanning, the detector receives the beam signals, which varies based on the intersected tissue’s density. If a high signal was obtained, this indicates a light tissue density (e.g. lungs) result in a dark colour, while getting a low level of energy implies a dense tissue (e.g. bones) with a bright white spot [11].

$$HU = 1000 \times \frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}} \quad (2.1)$$

Where $\mu_{water} = 0$ is the linear attenuation coefficients of water while $\mu_{air} = -1000$ is the linear attenuation coefficients of air. Each coefficient μ is an attenuation value of the beam for a given voxel, where a linear estimation is applied to compute the HU density value of the associated material as shown in equation 2.1. Any intensity less than water will be usually estimated with negative values in contrast to the denser structures, which will have positive values [12, 55]. Modern cross-sectional CT scanners can compose high-resolution volumes (e.g. commonly 512×512) with small pixel spacing (e.g. $.4 - .7mm$) and slice thickness (e.g. $.625 - 3mm$).

Substance	Hounsfield Units (HU)
Air	-1000
Lung	-600 to -700
Fat	-90 to -50
Water	0
CSF	+15
White Matter	+20 to +30
Grey Matter	+37 to +45
Kidney	+20 to +45
Liver	+55 to +75
Blood	+55 to +70
Bone	+300 (cancellous bones) to +3000 (dense bones)

Table 2.1: Some Hounsfield numbers for different human substance [11, 12].

MRI is another well-known diagnostic imaging modality that uses strong magnetic fields and radio waves to produce detailed images of patients. These medical images produced soft-tissue contrast for examining the patient brain and other internal organs, such as the liver, heart, and blood vessels, without harmful ionizing radiations [12, 55].

2.2.2 Analysis and Preprocessing

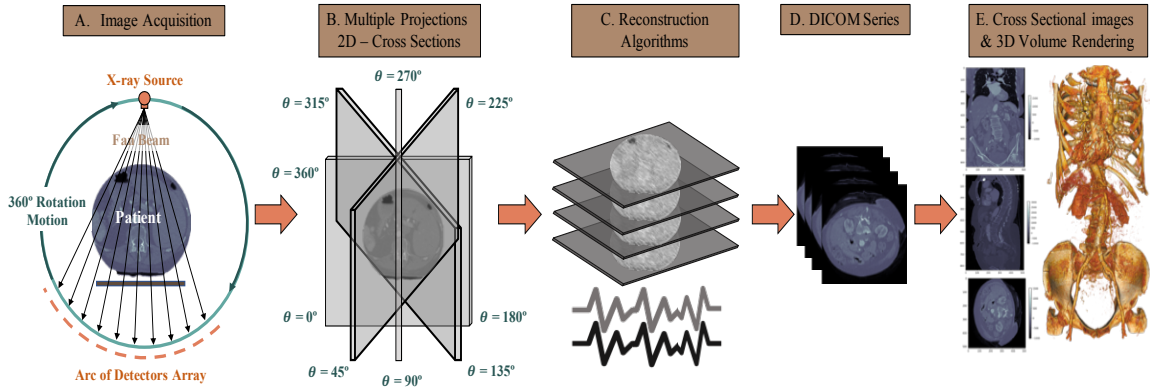


Figure 2.3: An illustration of the complete pipeline for processing and reconstructing 3D CT data.

A complete pipeline for collecting, processing and reconstructing 3D CT data is illustrated in Figure 2.3. The workflow begins with a data acquisition process through scanner detectors, where a progressive scanning of a patient's body is applied. These raw data are acquired and proceed to the next step, where several projections of the 2D cross-sectional from various angles are processed. Numerous algorithms could be applied in the image reconstruction phase to reconstruct the medical images and enhance their quality through noise and artifact reduction. The Filtered Back Projection (FBP) and Iterative Reconstruction (IR) algorithms are examples of some well-known algorithms [56, 57]. After reconstructing, a stack of cross-section images "slices" are produced and stored in a Digital Imaging and Communications in Medicine (DICOM) format series. These DICOM files can broadly be transmitted, stored, and processed within clinical purposes for many applications, including diagnosis and prognosis. Commonly, radiologists would gain insight into these volumes using volume rendering techniques to further analyse and explore these medical images.

Digital Imaging and Communications in Medicine (DICOM) is a recognised global standardisation for formatting, processing, managing and transmitting medical images and their related information for clinical objectives. DICOM file format consists of a header and the file meta information. The header is represented by the first 128 bytes followed by 4 bytes with the ASCII code characters 'DICM', while the meta-information portion comprises of four hierarchy levels (e.g. "Patient", "Study", "Series", and "Instance"). Within each hierarchy level, some related attributes are stored [58, 59].

Each metadata encoded information of a given Attribute, including a Tag, Value Representation (VR), Value Length (VL), and Value Field (VF). A unique identification tag consists of two primary parts four-digits defines group number and another four-digits for the element number, e.g.(gggg,eeee). The value representation is an optional attribute, storing two characters that specify VF’s datatype. On the other hand, the Value Length attribute is a mandatory field that defines the Length of VF. Lastly, the value field is the field where the actual data is stored [58,59].

DICOM format facilitates access to several modality types such as X-ray, CT, MRI, and ultrasound. More metadata about the patient’s medical information and scanning parameters is stored, providing compatibility for future potential analysis and investigation. Within the “Patient” hierarchy level, information such as the patient’s name, ID, age, sex is saved. Under the “Study” level, information related to the study ID, date, and time is included. The series number and modality type are attributes under the “Series” level. Other parameters related to the scanning details are slice location, pixel spacing, bits stored, slice thickness, scanning rows, and scanning columns. The pixel spacing attribute is a couple of two numbers that specify the physical distance in mm calculated between the centre of pixels horizontally and vertically. Another numerical attribute measured in mm is the slice thickness, which defines how thick each nominal slice is (i.e. the width of the human body covered by each slice) [59]. A single scanned image is stored as a stream of bytes in a DICOM file. Similarly, when aiming to store multiple cross-sectional images, each slice usually needs to be stored in individual DICOM files.

When retrieving CT image data from the DICOM file format, a preprocessing step is necessary to convert the data range to the HU scales. Initially, the medical images were saved into this format to enable efficient disk storage. A linear transformation is applied to rescale each raw voxel value back to the Hounsfield Units to guarantee the same radiodensity scale across all scans. From the DICOM file, two main attributes, namely RescaleSlope and RescaleIntercept, are required to mathematically reverse the data range using the linear equation 2.2. The RescaleSlope and RescaleIntercept typically have values of 1 and -1024 based on the DICOM standard from Siemens [60,61].

$$HuPixel = RawPixel \times RescaleSlope + RescaleIntercept \quad (2.2)$$

2.2.3 Visualization

“The greatest value of a picture is when it forces us to notice what we never expected to see.”

—John Tukey, American Mathematician

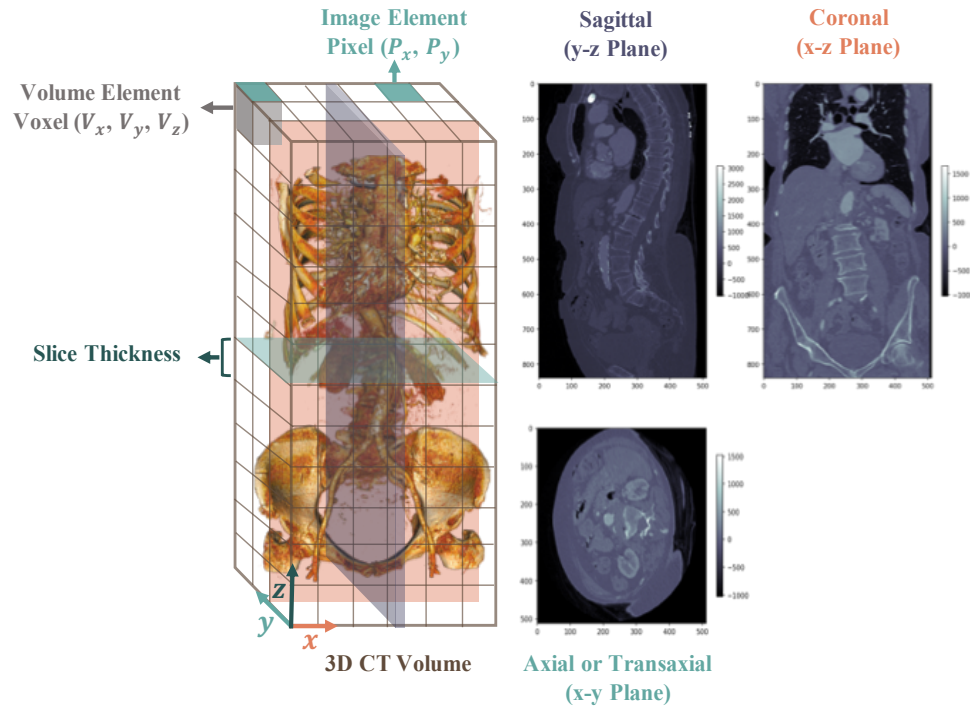


Figure 2.4: Illustrations of volume rendering (VR) of 3D CT volume and multiplanar view of three thin slices in the axial x-y plane (middle bottom), sagittal y-z plane (middle top), and coronal x-z plane (top right). The visualisation includes an example of volume element (voxel) and picture element (pixel).

Data visualisation is the science of revealing, exploring, and understanding the underlying data through visual representations. Gaining insight into the data is an essential step in analysing massive amounts of information, making data-driven decisions, discovering implicit knowledge, and amplifying the cognition of the data. Thus, visualisation applications are limitless, including fields, types of data, and phenomena. For instance, visualisation methods are necessary for the medical imaging domain to examine injuries, diagnosis, doctor-patient communications, treatment planning, and preoperation planning. The intentions of visual designs can further be utilised to emphasise the regions of interest, highlight the critical structure, and reveal hidden regions. Exploring high dimensional medical datasets can naively be done

by exposing slice-based viewing or cross-sectional multiplanar views across the 3D volume as illustrated in Fig. 2.4. However, such illustrations are limited in interaction and do not fully utilise the whole dataset as only subsets of the data is presented [1]. More complicated and visually distinguishable approaches include volume rendering, iso-surface rendering, or multimodel fusion [1, 62, 63].

2.2.3.1 Volume Rendering

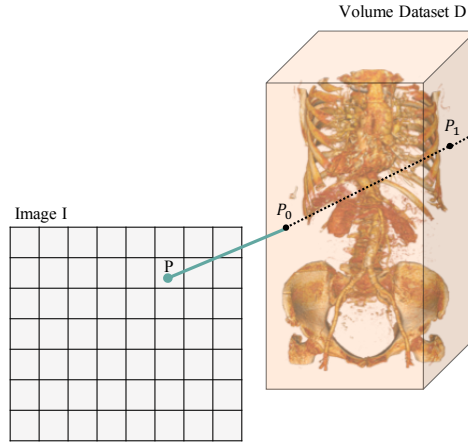


Figure 2.5: Conceptual principle of volume visualization [1].

Volume rendering or volume graphics is one of the state-of-the-art methods in visualising 3D medical volumes. The main objective of this rendering is to amplify cognition by interactively exploring the 3D scalar dataset in 3D space. The final image is computed as composited colour from the selected view angle using the ray casting technique, whereas a ray will be cast through the volumetric data from each pixel as illustrated in Fig. 2.5. During rendering, the pixel colour is accumulated along each viewing ray while moving through the volume (see equation 2.3). The colour range is assigned based on an optical model known as the Transfer Function (TF), which defines the optical properties (e.g. colour and opacity values) for each scalar belonging to the data [1, 63]. Some examples of rendering three different volume datasets are illustrated in Fig. 2.6.

$$\begin{aligned}
 C_{des} &\leftarrow C_{des} + (1 - \alpha_{des})C_{src} \\
 \alpha_{des} &\leftarrow \alpha_{des} + (1 - \alpha_{des})\alpha_{src}
 \end{aligned}
 \tag{2.3}$$



Figure 2.6: Examples of volume rendering of three different 3D medical image datasets.

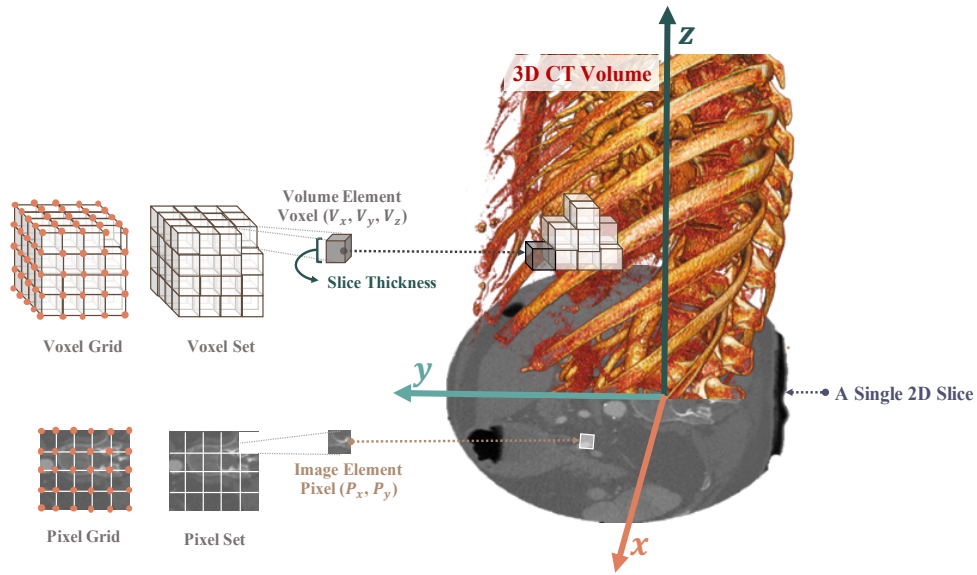


Figure 2.7: A visual overview demonstrates the difference between a volume element (voxel) and a picture element (pixel). An image comprises a set of discrete pixel (colour) at each 2D grid location, while volume is formed as a set of the discrete voxel at each 3D grid position.

As a pixel is defined as the smallest unit (picture element) in image space, in higher dimensional space (e.g. 3D volume), the smallest unit is known as a voxel (volume element). A voxel is defined as a grid point located with three axial components in the volumetric space. Each grid point may contain single or multiple data values associated with it. A collection of voxels or grid points form a cell. Commonly, in 3D CT volume, each grid point represents an intensity value (e.g. a colour). Both 3D volume and 2D image have a discrete value at each grid location; however, if a value

in an intermediate position is needed, an interpolation operation can be applied to the data points at a neighbouring area to obtain those particular values. An illustration of the difference between a picture element (pixel) and a volume element (voxel) is demonstrated in Figure 2.7.

When examining medical images, the visual coordinate system utilised by radiologists and doctors is known as an anatomical coordinate system, defined along the standard anatomical axes of anterior-posterior (i.e. front-back), left-right, and inferior-superior. This coordinate system, also known as the patient coordinate system, consists of three planes (i.e. axial, sagittal, and coronal planes). However, the voxel-based coordinate system is formed differently as a grid (i.e. array of points) in 3D, whereas the origin is defined in the upper left corner point (i.e. first voxel (0, 0, 0)). Moreover, each axis increases in a different direction (i.e. the x-axis increases to the right, the y-axis to the bottom, and the z-axis backwards) [59,64,65]. This thesis will visually illustrate all the medical scans from a machine learning perspective (i.e. in the coordinate system of voxel space) as the data is analysed and processed, and not in the patient coordinate system (i.e. anatomical coordinate).

2.2.3.2 Transfer Function

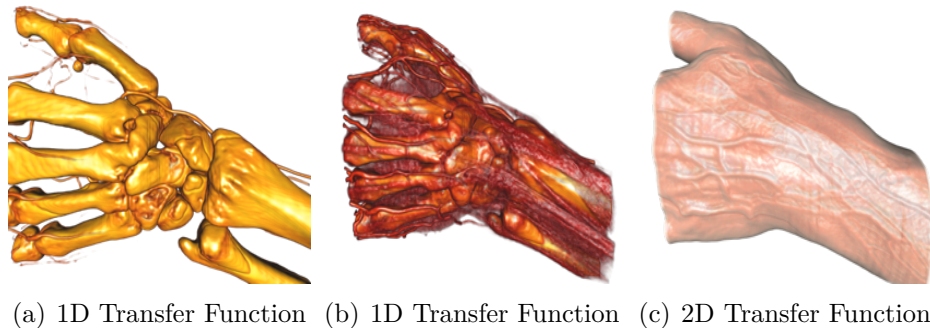


Figure 2.8: A demonstration of rendering the same dataset using different transfer functions, whereas each mapping visually classifies different features of interest.

A transfer function is a core to a visually distinguishable appearance, where a mapping from the scalar field values to the optical properties (e.g. RGBA values) is applied. The optical properties consist of colour components (i.e. RGB) and visibility component (i.e. A), representing the opacity [1]. Different mappings would visually classify features of interest, reveal diverse materials, and focus on various aspects of the underlying data as demonstrated in figure 2.8. Such numerous options cause selecting

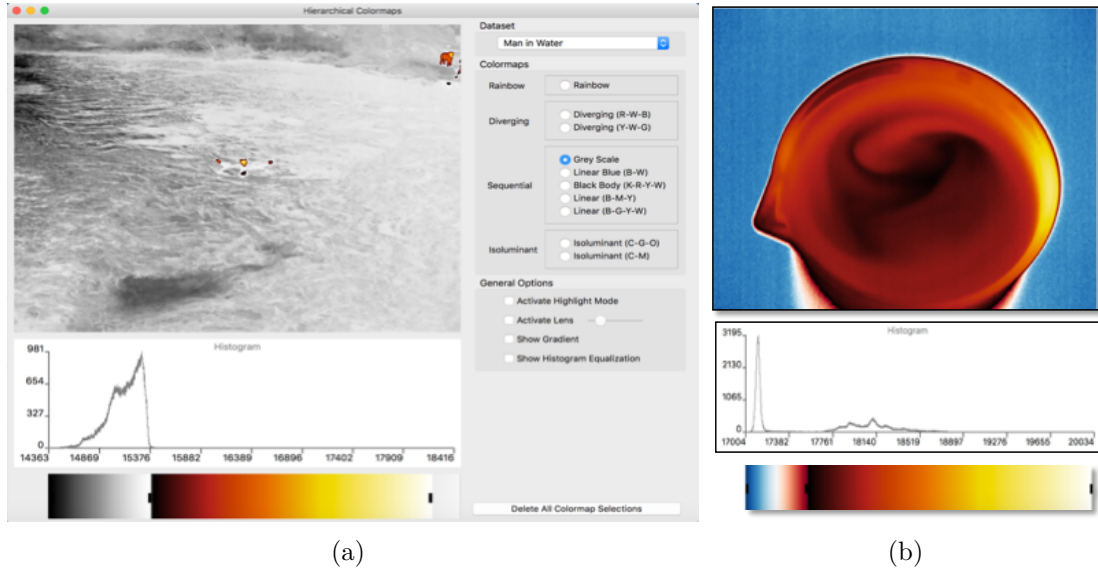


Figure 2.9: The main interface of DataPainter 2.9(a), showing the image derived from the colourmap, the histogram of the thermal image (640×480), and the current user colourmap option. Another example of rendering a thermal image dataset (i.e. hot cup) is illustrated in 2.9(b), whereas the nested colourmap emphasises the extensive dynamic range of the underlying data [52].

a representative transfer function quite challenging and require manual efforts by specialists. Consequently, a considerable amount of efforts have been made to semi-automate and improve transfer function design, including computing high-dimensional TF [66], involving deep learning techniques to assist the designing of TF [67]. The mapping function's dimensionality can compose one or more dimensions combining intensity values with more data-driven data features (intensity-gradient) such as the first derivative (as shown in Fig. 2.8(c)) and the second derivative [66]. Other complicated classifications utilising 3D filtering to highlight local structures (e.g. edge, line, and blob) involve gradient vector, and the Hessian matrix with isotropic Gaussian blurring [68].

The use of a transfer function is not reluctant to render medical domain imageries only; it can extend to cover a broad range of data domains. The colour mapping or colour lookup table is an essential part of the optical priorities utilized by the transfer function. Based on the colourmap selection, the resulting image can highlight certain information about absolute or relative data values. A novel and interactive tool for detecting features of interest using nested colourmaps was proposed by Nagoor et al. [52]. Such a technique can be applied as a diagnostic tool for building issues, namely, detecting faulty cables (hot), water leaks (cool), problems with air permeability or

other insulation issues. Data Painter is a tool that provides a framework for effectively defining customized colourmaps, which maximize the user understanding of the underlying data. The user can sweep colourmaps across the image in order to reveal hidden features and expose essential details, as shown in the examples in Figure 2.9. Work on “Data Painter: A Tool for Colormap Interaction” was originally published in the Computer Graphics & Visual Computing (CGVC) [52] in 2017, by the thesis author alongside Dr Rita Borgo and Prof. Mark Jones. This paper awarded The Best Full Paper Award.

2.2.4 Challenges and Observations

- There has been a knowledge gap between domain-specific experts (e.g. radiologists and medical experts who have working experience with hospital scanners and scanning settings) and the developer or technology provider (e.g. computer scientists and programmers who understand and apply deep learning knowledge). This gap is due to the lack of communication, which generally affects knowledge transfer and overall understanding, causing the development of AI strategies to be time-consuming and less accurate. Collaboration and iterative communications can potentially address this gap by leveraging the information, improving overall quality, accomplishing the assigned tasks better, and helping AI methods be integrated into the clinical workflow [69–71].
- Nonuniform data productions are due to variations in scanning qualities or settings (e.g. different data may be generated depending on how modern a scanner is or when changing some scanning settings). As the scanner’s beam settings highly influence the HU’s ranges, any variations in beams’ energy would impact attenuation values, making this one of the technical challenges related to the HU. Such a problem would cause generating relative ranges rather than absolute, causing overlap between HU (CT) ranges of different organisms [55, 72].
- Other challenges related to volumetric imaging is noise and artifacts occurrence. Noise, intensity inhomogeneities, patient motion artifacts and partial volume effects are examples of artifact problems that may appear within multiple medical imaging domains and generally influence the accuracy of the computational methods. Particularly, partial volume effects result in blurring intensities at tissue boundaries due to multiple materials contributing to a single voxel. Such scanning artifacts cause averaging values of these tissues to the same voxel, leading to obscure details or nonuniformity of a single material across different parts of an image. Moreover, these limited resolutions could lead to misdiagnosis or limit algorithms’ accuracy due to difficulty in characterizing individual tissues when solving various tasks (e.g. segmentation, visualization, or coding tasks) [73–75].
- The trade-off between scanning quality and emission doses on the patient’s body (i.e. radiation dose that would be risky affecting the patient body). Such a critical issue could be managed by developing optimisation methods and guidelines that tend to maximise image quality at minimal radiation dose [55, 76–78].

2.3 Data Compression

The ever-increasing importance of accelerated information processing, communication, and storing are major requirements within the big-data era revolution. With the explosive development in storage devices, Graphics Processing Units (GPUs), and bandwidth capacity, the data growth rate also rises, demanding more practically accelerated yet efficient algorithms. Data compression is one of the most vital fields of computer science, providing methodologies for compressing information into a more compact form. Data compression applications are numerous, including numerous digital devices (e.g. smartphones, or TV), media and data representations (e.g. videos, audios, or images), and various data communications and streaming domains (e.g. Internet and Network communications) [79].

2.3.1 Compression Basics



Figure 2.10: A general block diagram of a data compression framework.

The term compression is naturally referring to the process of reducing data size or bit rate by utilising some statistical or structural pattern in the given raw format. A general block diagram of a data compression framework is presented in Fig. 2.10. The scheme at which compression is applied can be classified as either a lossless or lossy type. Briefly, a lossless scheme guarantees exact reconstruction of the original data, while a lossy scheme would generally allow a higher compression ratio but non-identical reconstruction of the original data. A comparison between the two different strategies is illustrated in table 2.2. Selecting which compression scheme to apply generally depends on the application requirements. A general classification of the data compression techniques is illustrated in Fig 2.11, where examples of coding categories and their compression techniques are presented. Some compressing algorithms combine different coders' functionality joined together (hybrid) to further reduce the given data size [80,81].

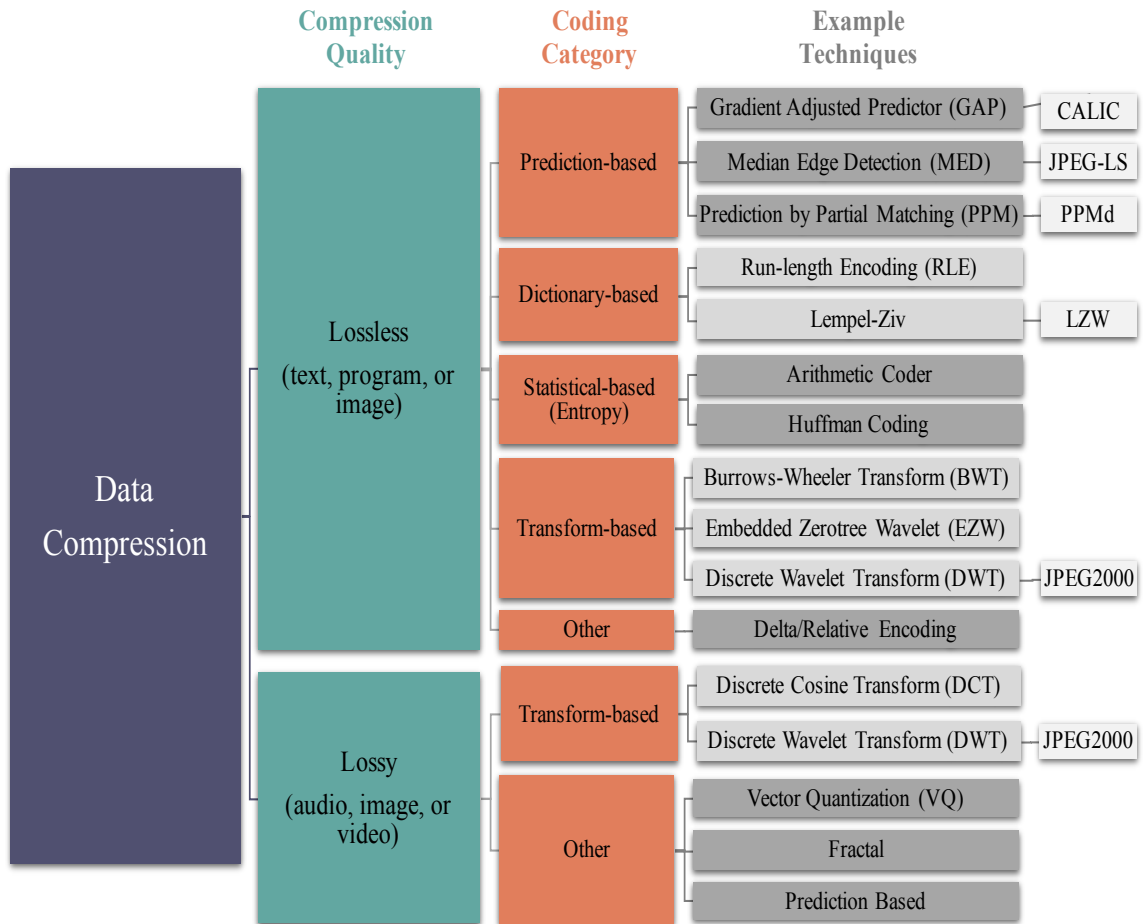


Figure 2.11: A general classification of data compression techniques.

Most compressing schemes utilise some structural, mathematical, and statistical information in the actual data when shrinking the file size. A compressor gains reduction by discovering, processing, and reducing the data redundancy within the raw format. The main intention is to exploit any patterns, repetitions, or correlations in the input data or transform it into a format that reveals such patterns to produce a more compressed form of the original version. The redundancies in any data can be classified into spatial, temporal, perceptual/psycho-visual, and coding redundancy. While each repetition has different characteristics and can be associated with different types and fractions of data, the size of the output representation decreases by encoding

	Lossy	Lossless
Compression Ratio	Lower bit-rates	Higher bit-rates
Compression Time	Usually fast	Usually slow
Quality of The Data	The reconstructed data has less accuracy since it involves irretrievably loss of the original information	Completely reconstruct the original image
Compression Artifacts	May suffer from compression artifacts such as blocky and blurry appearance due to the loss of original information while compressing	No compression artifacts
Assessment Metric	Needs assessment metrics to measure the perceptual quality of the reconstructed image, i.e. Structural Similarity Index Metric (SSIM), and Peak Signal-to-Noise Ratio (PSNR) [82, 83]	No assessment metric needed since the reconstructed image is identical to the ground truth (GT)
Application Examples	Image, audios, or videos	Text, programs, images, or sounds
Algorithm Examples	Transform coding, DCT, and DWT [84, 85]	Lempel-Ziv Algorithm, LZW [86–88], Arithmetic coder [89, 90], Huffman coding [91], and Run-length coding [92]

Table 2.2: A general comparison between Lossy and Lossless compressing schemes.

the redundancies.

The **spatial** type represents the statistical correlations or repetition in signal or block of values within a domain or structure, for example, similar pixel values repeated in many blocks (e.g. neighbouring pixels) within a Two-Dimensional image. Such redundancy is sometimes referred to as interpixel correlation, which naturally has similarity and dependency between neighbouring pixels. A coder that exploits spatial redundancy usually applies a prediction scheme in which the current pixel’s residual (i.e. prediction error) is encoded given a block of its neighbouring pixels. When pixels are highly correlated, they naturally produce recurring patterns within the same image, allowing the coder to infer current value from its associated surrounding values. The term residual often indicates the difference between the original data value and a compressor’s prediction value. Removing spatial redundancy creates a more efficient

format widely employed in many well-known image and video compressors.

The **temporal** redundancy is similar to the spatial repetition but extends to exploit statistical correlation within 3D data or frames in a video sequence; thus, it may also be known as interframe redundancy. In this type, compressors apply a prediction scheme to decorrelate pixels between successive frames leading to efficient video compressions.

The **perceptual/psycho-visual** redundancies are related to a particular type of multimedia data such as images and audio. Such types take advantage of a Human Visual System (HVS) or hearing sensitivity level while compressing the given data. Within such type, visually indistinguishable colours or indiscernible sounds values are removed to reduce file size while essential parts are preserved. Since certain information is psychovisually less relative and would not significantly affect the overall perceived quality of an image thus, it can be safely excluded. The same principle is applied to sound compression, where inaudible frequency bands are reduced to produce a more compressed and efficient version.

A **coding** redundancy is a type in which the repetition is revealed on a bit level (the information representation) by a coder. An Entropy coder can be operating as a fixed-length or variable-length coding scheme while estimating the likelihood of the repeated bit sequences. In variable-length coding, the high-frequent values are assigned shorter code while less frequent values are assigned longer code. Huffman coding and arithmetic codes are examples of some standard Entropy codes (covered in detail in section 2.3.2.4).

2.3.1.1 Lossy Data Compression

Although some part of the original data is irretrievably getting lost within lossy compression, the most critical parts are usually preserved. For instance, compressing an image with a lossy compressor (e.g. perceptual coding) may lead to losing some visually indistinguishable colours while the essential parts and structures will still be saved [93]. A similar reduction principle is applied in some lossy audio compression methods, where inaudible frequency bands are compressed by exploiting the properties of human hearing on indiscernible sounds [94, 95].

2.3.1.2 Lossless Data Compression

Regardless of the compression category or quality, the central objective is to reduce the storage space or the transmitting time required for handling the raw data format. In

a lossless scheme, the compressed representation has reduced the number of bits while the algorithm ensures recovering the original form of data when decompressing as losing any details cannot be accepted. Such a restriction is demanded in some domain areas where the content is significantly important and cannot afford to miss any fraction of it because any modifications could affect information-preserving, integrity, and reliability. For instance, the lossless strategy is more appropriate in the medical image domain since it recovers the original data without losing quality. As within the medical field, any variation between the original and reconstructed data usually cannot be tolerated since it affects the data fidelity and quality (e.g. misleading diagnosis). Text compression is another application that demands identical decompression of the original information as any loss in quality would affect the data integrity [96]. Further applications that require losslessly encoding and decoding information includes satellite imaging, program, and sound compression [51, 79].

2.3.2 Compression Algorithms

2.3.2.1 Shannon Entropy

Information theory quantifies the amount of information present in random variables and distributions. Although this theory has initially been used to measure information in message communication, nowadays, it forms the core of many language analysis and deep learning algorithms. Data compression is a field of information theory interested in techniques for reducing the size of a data stream to overcome the bottleneck of data storage, transmission and processing. In the context of data compression, information theory is applied to find the optimal codec that achieves the best compression and the lowest entropy. Entropy estimates the minimum number of bits required on average to represent symbols of a stream. Based on information theory, a symbol with a higher probability is assigned the shortest code (less informative), and the less probable symbols are assigned longer codes (more informative). As a result, a lower entropy is gained from nonuniform distribution since higher frequency symbols or events have shorter coding lengths. The more deterministic the data distribution is, the lower entropy the coder gains [97–99]. The measure of the uncertainty of a probability distribution P is known as Shannon entropy H , which is defined as:

$$H(x) = -E_{x \sim P}[\log P(x)] \quad (2.4)$$

Lossless compression is known as reversible compression, which aims to send a stream of data m with a codelength bounded to the entropy $L(m) \geq H(m)$. Given a

message $m \in M = \{m_1, m_2, \dots, m_N\}$ with probability distribution $P_{data}(m)$, a *sender* will encode the data m into a sequence of bits with a codelength $\hat{L}(m)$. The encoder uses a probabilistic model $\hat{P}(m)$ to define the codelength $-\log \hat{P}(m)$ for each symbol m_i in the stream. If the codelength $l(m_i)$ is equal to $-\log P_{data}(m_i)$ for all symbols in the message, the encoder achieves the maximum compression (entropy):

$$H(m) = - \sum_{m_i \in M} \hat{P}(m_i) \log P(m_i) \quad (2.5)$$

When the *receiver* gets the compressed bitstream, a decoding or reverse operation is applied and the original data m is fully reconstructed without any loss of information.

2.3.2.2 Prediction-Based Coding

2.3.2.2.1 Prediction by Partial Matching Coder A Prediction by Partial Matching (PPM) is one of the statistical-based (entropy) and context-based models, where the previous sequences in the context are utilized before the following symbol's probability is predicted [100, 101]. The PPM coder is a dynamic method that exploits the dependencies in a stream by assuming the probability of unseen symbols based on the probabilities of the symbols that have been seen. A Markov model is employed to conditionally predict the probability of the next element in a sequence (also known as a context) based on the immediately preceding symbols. In this algorithm, n specifies the largest number of previous symbols to be seen before a prediction is performed, also known as the model's order, denoted as PPM(n). The PPM algorithm employs a counter to encode the number of times a symbol was detected. This counter can remember some of the contexts as it adaptively calculates the number of times a symbol occurs at each step based on the given order- n . The symbols' counter values are stored in a table, where each row represents a symbol from the stream with its counter value. When coming across a symbol observed before then, the coder encodes its counter and updates the table. To estimate the probability of a specific element, its counter's value will be divided by the summation of contexts' values along with the value of a particular character known as an escape. The escape ϵ symbol is a unique character used to denote the probability of a never-seen symbol. When the ϵ symbol is generated, the encoder will try a smaller context (e.g. $n - 1$). If no match was found after reducing the context, then repeatedly reduce the context until a context of order -1 will eventually be produced, indicating an unseen or new symbol. The resulted data distribution estimations with their tables would then need an entropy coding method to reduce their coding redundancies. Several variations of the PPM

coder have been produced, whereas each version may differ in the methodology for computing the counter value and the ϵ symbol codeword value. A popular version is the PPMd [102], which is one of the compressors used within the 7-Zip archiver [103].

2.3.2.3 Dictionary-Based Coding

2.3.2.3.1 Run-Length Coding A Run-Length Encoding (RLE) is considered one of the lossless dictionary coding algorithms, where the repeated sequences are encoded as pairs of string lengths (called a run) with their symbols [92]. The principle applied in RLE is replacing long iterated chains with more compact representations that encode both the number of repetitions and the symbol. Although RLE is considered one of the simplistic encoding algorithms, its methodology may cause the compressed file's size to be relatively more significant than the original size in a worst-case scenario. The RLE is commonly used to reduce spatial redundancies in data representations such as text, string and image.

2.3.2.3.2 Lempel-Ziv Algorithm Lempel-Ziv algorithms are a group of well-known lossless compression algorithms extended for decades, where the original versions (*LZ77* and *LZ78*) were initially introduced in 1977 and 1978 by Jacob Ziv, and Abraham Lempel [86,87]. Most popular lossless dictionary compressing methods such as *LZW* [88], *LZMA*, and *LZSS* [104] are variances of these two original versions in which slight modifications to the buffer searching procedure and the way of outputting the token are utilised. The main mechanism is building a compressed dictionary (i.e. data structure) that is adaptive to the input stream.

The *LZ77 dictionary-based compressors* rely on a sliding window principle, splitting an input stream into search-buffer (SB) and look-ahead (LA), where the search-buffer contains symbols that have been encoded while the look-ahead buffer has elements that have not been observed yet [86]. An overview of the *LZ77* lossless compression algorithm, is presented in Fig. 2.12. The sliding portion principle is an adoption scheme that reduces any symbol's redundancy by pointing to its first appearance. While processing a stream, the compressor reads a symbol from the LA buffer and searches for a match in the SB buffer. When a match is found, the encoder will read another symbol from the LA buffer while searching backwards for a more significant match in the SB until it finds the longest match. If the longest correspond is located, the encoder will output a token containing three main components [offset, length, and next token in the stream]. A token is practically a pointer to the earliest occurrence of the matched element. If no match is found (a new element), the

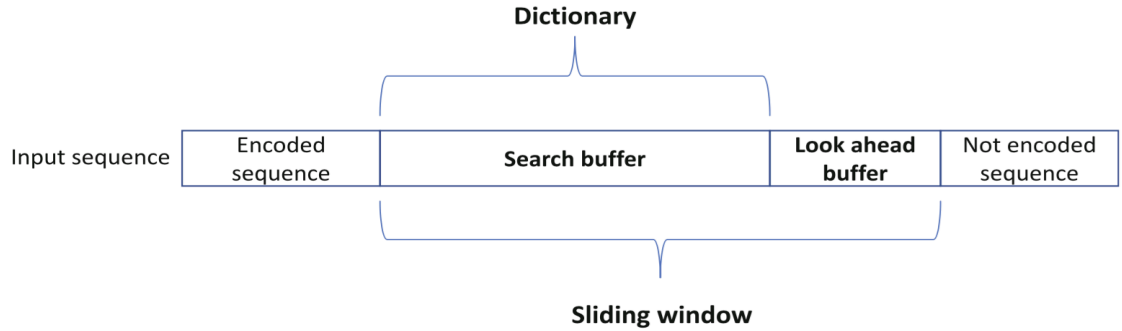


Figure 2.12: Overview of LZ77 lossless compression algorithm [2].

output token will be a “no token” or “null-pointer” (e.g. [0,0, the token itself]). The compressed version of the data will be a list of these output tokens.

In the *LZ78 dictionary-based compressors*, the output codeword consist of a pair of only two components, including an index to the longest matching and next unmatched symbol [87]. The fundamental methodology of *LZ78* methods is building a dictionary by storing the list of the output tokens. Similarly to the *LZ77*, when no match is found, the output token will be a “no token” (e.g. [0, the token itself]). The main advantage of the *LZ78* is its fast encoding time compared to the *LZ77* method, but in terms of decoding time, the opposite is true. The main limitation of the *LZ78* method is the failure to limit the dictionary size.

Terry Welch invented a variation of *LZ78*, which is **Lempel-Ziv-Welch LZW** in 1984 [88]. The *LZW* maintains a running variable-length dictionary of symbols, which adapted a dynamic codeword table for each file while continuously optimising for the longest possible match. Similarly to the *LZSS* algorithm, *LZW* excludes the next non-matched symbol from the generated token and only stores a pair of [the offset, and length]. The main weakness of *LZW* is the limited ability in handling table overflow.

The **Lempel–Ziv–Markov chain algorithm LZMA** is a well-known dictionary algorithm that compresses various data representations losslessly. LZMA is the backbone behind the popular 7-Zip archiver or 7z format developed by Igor Pavlov around 1998 [103]. The LZMA belongs to the family of the LZ77 algorithm, which uses the Sliding window mechanism to maintain a dynamic size dictionary compression. Similarly to the other *LZ77* algorithms, an encoding of [0,0, the new symbol] will be used when no identical bytes are found. Moreover, the LZMA is considered a variation of the Deflate algorithm combined with the Markov chain, where the Delta

and Range encoders are employed. Although the LZMA compressor outperforms other compressors in terms of compression ratio, its main disadvantages are that it requires substantial memory consumption, and the implementation might be relatively slow depending on the input data. An enhanced and parallelized version of LZMA known as LZMA2 is widely used in many domain applications and included in the 7-Zip archiver [103].

2.3.2.4 Statistical-Based Coding

2.3.2.4.1 Huffman coding Huffman coding is one of the most popular lossless compression algorithms widely applied for text compression and was initially invented by David Huffman in 1952 [91]. Huffman coder relies mainly on the variable-length coding concept on which a binary tree is built to uniquely define an optimal binary code for each symbol belonging to the string based on the likelihood of its occurrence. The Huffman tree is constructed in a bottom-up manner, whereas words with the highest probabilities are assigned shorter bits (usually located at a high level of the tree), while words with less probabilities will have longer codewords (usually placed at the bottom of the tree). Each symbol will be assigned a uniquely distinguishable binary codeword with a distinctive prefix-code, making the data more compressible while avoiding ambiguity in the decoding process. Many variations of the original Huffman algorithm were presented, for instance, the Adaptive Huffman coding. An example of constructing a Huffman Tree for the symbols list $S = \{a, b, c, e\}$ is presented in Figure 2.13.

Creating a Huffman tree starts with a list of symbols along with the frequencies of their occurrence. The process started by sorting the list's elements based on their likelihood from the highest to the least. Then, select two of the lowest weighted elements as leaves and add a new parent node weighted by the summation of probabilities from these two child nodes. Next, remove the two processed symbols from the list and repeat the same process until no symbols are left in the list. After constructing the Huffman tree, each branch will be assigned a binary code, whereas left branches are assigned zeros while the right branches are assigned one. Finally, to gain the codeword of any symbol, traverse the tree directly from the root to the symbol's node while connecting the bit code of branches to create the unique code.

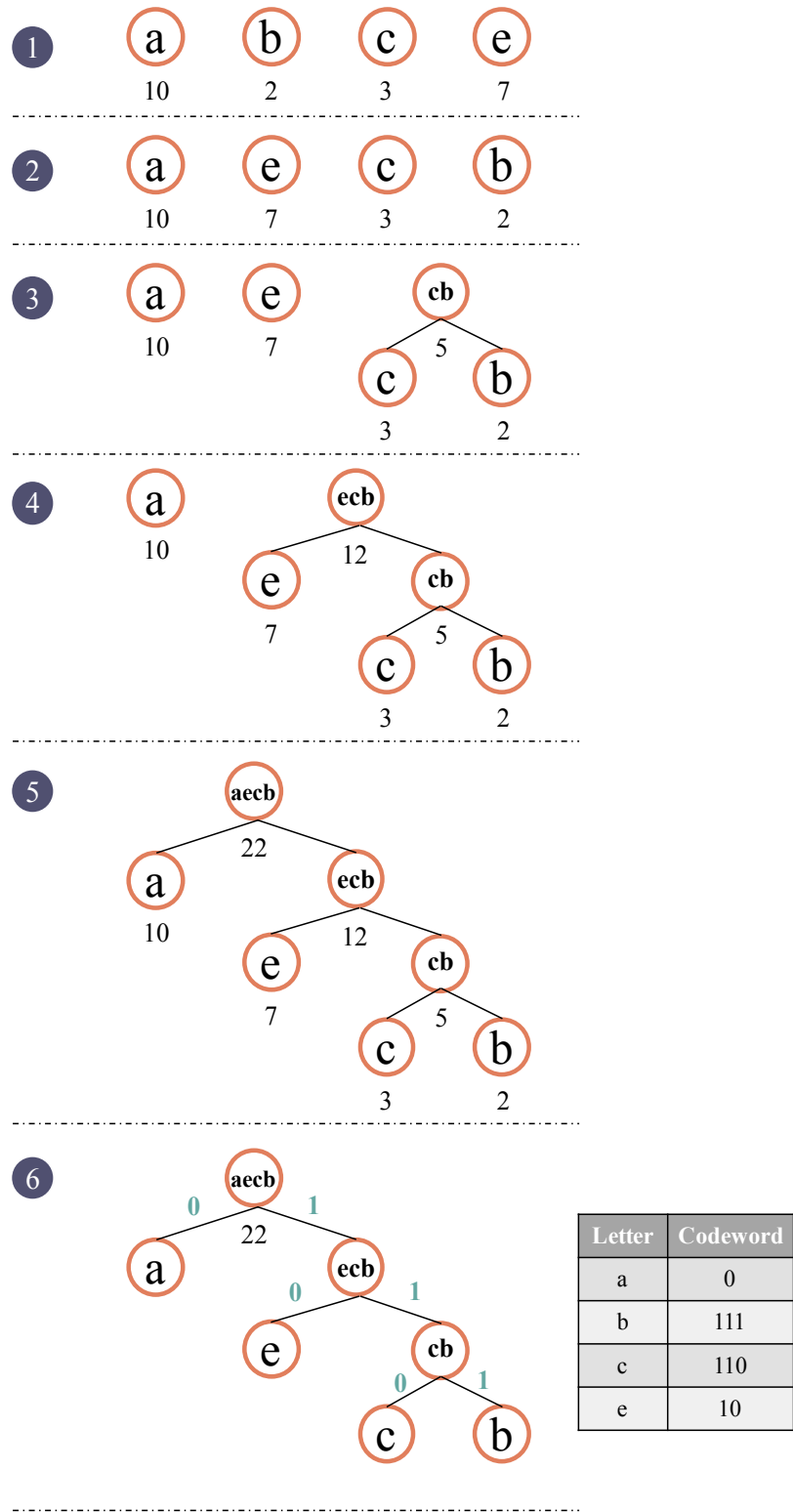
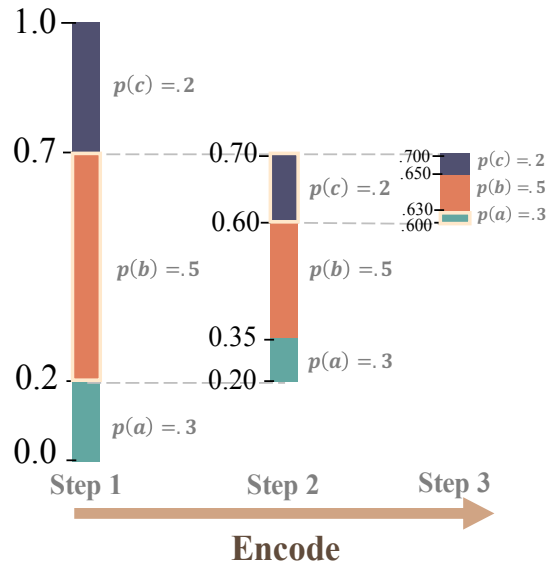


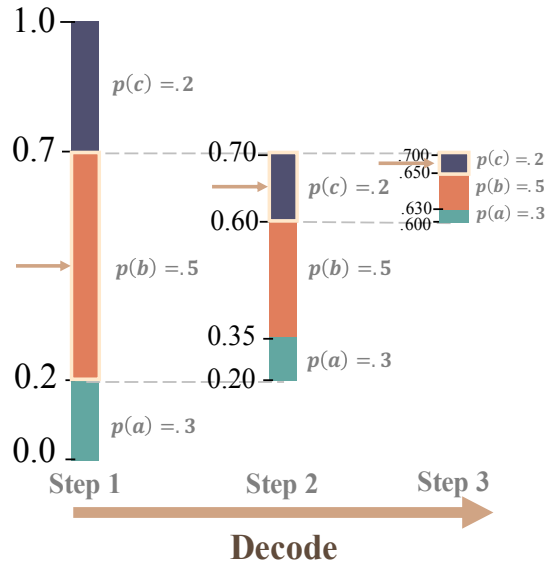
Figure 2.13: An example of constructing a Huffman Tree for the symbols list $S = \{a, b, c, e\}$.

2.3.2.4.2 Arithmetic Coder An arithmetic coder supports assigning intervals of floating values for each possible input symbol. This statistical-based compressor was initially developed in the early 1960s by Peter Elias [105] and has been significantly enhanced since then [89,90]. The arithmetic coder overcomes the limitation of others VLCs for defining a codeword per input symbol by specifying a single floating value to be the code for a range of input symbols instead [106]. Such a mechanism allows reducing the coding space to the range of 0.0 to 1.0 values, where each symbol will be assigned a unique subsection based on its frequencies at each step. During the encoding process, the active symbol's section will be subdivided into new smaller subsections based on the statistical probabilities. Accordingly, the starting and ending limits for the new interval will be bounded to the currently selected section scales. The algorithm will recursively continue partitioning based on the input probability distributions until no symbol is left in the stream. Usually, the more probability a symbol is, the less digit its code becomes, making the compression stream close to entropy. By subdividing intervals into more spaces and flexibly adding new floating ranges for any symbol in a stream, this coder can be adapted to different data types. Moreover, given that the interval's of each input symbol is uniquely defined, such data transformation functionality is one of the main advantages of the arithmetic coder.

An example of using Arithmetic Coding to encode a message sequence $M = "bca"$ is illustrated in Fig. 2.14(a), where $A = \{a, b, c\}$ is the set of alphabets, $p = (0.3, 0.5, 0.2)$ is the probability distribution for each symbol, and $[.6, .63)$ is the resulted sequence interval. The decoding follows the exact subdividing mechanism; however, the process starts from the symbols' interval when decoding. Given an interval value of (e.g. $.66$), an arithmetic decoder will continuously divide the section where the decimal value is located while revealing the message alphabets. An example of using Arithmetic Coding to decode the message sequence: $"bcc"$ is illustrated in Fig 2.14(b). The decoding process starts from a unit interval value of $.66$, and by considering three-letter alphabets $A = \{a, b, c\}$ with the probability distribution for each symbol given by $p = (0.3, 0.5, 0.2)$, the output message $"bcc"$ will be revealed.



(a) Encode a message sequence $M = "bca"$. The final sequence interval is $[\cdot6, \cdot63]$.



(b) Decode a message sequence of $M = "bcc"$ starting from the interval value of $\cdot66$.

Figure 2.14: Example of using Arithmetic Coding encoding and decoding process, where consider a three-letter alphabet $A = \{a, b, c\}$, and the probability distribution for each symbol is $p = (0.3, 0.5, 0.2)$. The encoded message sequence is $M = "bca"$ while the decoding process started from the unit interval value of $\cdot66$, and the resulted message sequence is $M = "bcc"$.

2.3.2.5 Transform-Based Coding

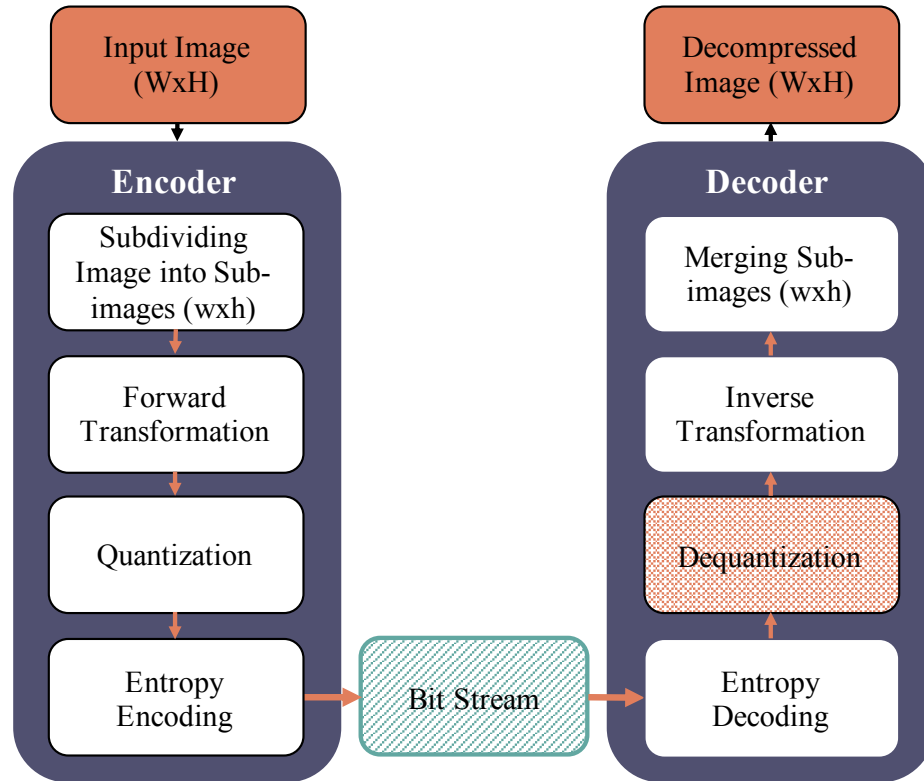


Figure 2.15: A general block diagram of a transform-based coding framework.

A transform-based coding is a mechanism that transforms data with nontrivial computation into another representation, which can be more compressible. When the transformation is reversible, this enables a lossless compression, while non-reversible modifications will produce a lossy compression as it loses proportions of the original data. The reformed format is sometimes referred to as transform coefficients, which are then quantised and coded. Most transform-based coders begin by subdividing the input data (e.g. an image) into smaller blocks to reduce computational complexity when applying the transformation. The applied transform-based method may adaptively differ for each subregion based on the local information and underlying variance, which tend to reduce each into a more compact form. Another option is the non-adaptive transform, which contains all the subregions to have the same compressing transformation. When the compression quality is lossless, a decoder will typically follow reversible operations corresponding precisely to the encoded block. Overall, the compression stages when employing a transform-based coder can generally be summarised to the following operations: subregion, decomposition, transformation, quantisation, and

coding. A general block diagram of a transform-based coding framework is shown in Fig. 2.15. The Discrete Wavelet Transform (DWT), the Discrete Fourier Transform (DFT), and the Discrete Cosine Transform (DCT) are some famous transform-based codings used in many domain applications, including image and signal compression.

Other usages are within image analysis, classification, filtering, and reconstruction scopes. More complex domain-specific applications can be image segmentation, matching, registration, object detection, and motion analysis. Numerous influential contributions utilising Fourier Transform while analysing the characteristics and structural patterns within images have been proposed, wherein decompositions, transformation estimations, and analysis of the waves and frequencies are applied [107, 108]. Briefly, an image's signal can be computationally described as alternative combined waves of various magnitudes, frequencies, and phases by transforming from the original spatial domain to the Fourier or frequency domain. In such representation, low-frequency components are generally located within the centre of the Fourier space while high-frequency elements (e.g. edges in the image) are at the boundaries of outer regions. Patterns and features can be detected and extracted through complex masking/cropping operations, spectrum filtering, or spectrum feature models/descriptors within such domain. More complicated yet enhanced detectors and advanced appearance models may involve subband filtering, multiscale wavelet appearance pyramid, or multiscale decompositions for better textures and features capturing. Considerably effective utilisation of these concepts is applied in medical image analysis, visualisation [68], and classification and segmentation tasks [109–113] to assist clinical and pathological studies.

2.3.2.5.1 Discrete Fourier Transform (DFT) A Fourier Transform measures a periodic function that can be decomposed in terms of infinite sine and cosine functions of various frequencies (may know as a set of Sinusoids) [114]. Generally, a Fourier transform calculates a correlation between functions or precisely a signal and an analytical function. The main intention is to calculate the coefficients corresponding to each frequency. The overall equation is composed of multiplication between a signal and an analysing function (Sinusoids). The resulting coefficients' values will be significant when the functions are correlated. However, if the multiplication result were negligible coefficients, the two functions would be dissimilar. In a Continuous Fourier Transform, the sine and cosine correlation coefficients are computed as two separate integrals for each function. On the other hand, for the Discrete Fourier Transform (DFT), the frequency coefficients are computed as a weighted summation over a finite

set of samples within the domain range as shown in equation 2.6. The output of DFT is a vector of estimated frequency components (Fourier coefficients) for each data point. A computationally efficient variation of DFT is the Fast Fourier Transform (FFT), which is gaining significant popularity in several compression applications, including various images and audios compressions [84].

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi kn}{N}} \\ X_k &= \sum_{n=0}^{N-1} x_n [\cos(\frac{2\pi kn}{N}) - i \cdot \sin(\frac{2\pi kn}{N})] \end{aligned} \quad (2.6)$$

Where N represents the number of samples, the k is current frequency within the range $k \in [0, N - 1]$, and n is the current sample. The X_k is the DFT frequency coefficients that includes information of both amplitude and phase.

2.3.2.5.2 Discrete Wavelet Transform (DWT) also known as multiresolution analysis, where signals are processed in numerous frequencies and resolutions levels. Compared to the other transform-based methods, DWT offers less computational complexity and is suited to compress time-limited data representations (e.g. images and audios). Generally, a Wavelet Transform interprets the input as a sum of wavelet functions on different resolution scales by which details coefficients represent high frequencies signal components while fewer coefficients represent low frequencies components. A wavelet is a tiny wave (base function) that performs a windowing mechanism to maintain input signals through scaling. When scaling a signal, both the signal's width and central frequencies are transformed, allowing expanded waves to resolve low frequencies while shrunk waves in case of high frequencies. In the Discrete Wavelet Transform, a multilevel decomposition is computed in which input signals are iteratively passed through low-pass and high-pass filters. Each level comprises approximated coefficients and dilated coefficients components. Such a mechanism allows rejecting high signals while proceeding signals at low levels. It also supports dilating and stretching signals while going across them. As a result, the DWT would lastly generate a set of both approximated and dilated coefficients. When compressing an image, the DWT maintained a better quality and more accurate results than its alternatives as it will not generate computationally significant data but rather efficiently extract multiscale and hierarchical structures [84].

2.3.2.5.3 Discrete Cosine Transform (DCT) the Discrete Cosine Transform belong to the unitary transform class, whereas it decomposes signals as a series of only cosine functions with oscillated at different frequencies [85]. Similarly to the DFT, a

DCT converts input signals from the spatial domain to the frequency domain resulting in a weighted sum of basis functions (or basis images). However, a DCT is only applied for Real numbers and approximates fewer frequency coefficients than DFT. Thus, DCT performs better in data compression, commonly in image compression domains. The applications of DCT is not reluctant to images only; it extends to include various data dimensionalities such as 1D, 2D, and 3D spaces. The process starts by subdividing an image into blocks on which each region may be decomposed with a separate basis matrix with different variations. After computing frequency coefficients, a quantisation process reduces coding redundancies and earns reduction. When a lossy compression is performed, only significant coefficients are preserved as they contain essential spatial frequencies (e.g. perceptually distinguishable features), while less critical information can be safely reduced. Such a mechanism forms the central core of the JPEG compressor. The same principle is applied when compressing audio information with a lossy compression quality, where inaudible frequency bands are reduced while more important frequencies are retained.

2.3.2.6 Quantisation

Quantisation is a mathematical procedure for decreasing the bit required by lowering input precision, wherein the continuous infinite values are mapped to fewer discrete finite values. A quantisation is generally applied to help regulate the bitrate within a bitstream to more compressible representations. The principle is that rounding the stream's range to the nearest discrete or quantum levels can result in a more compact format. However, as the approach involve losing some precisions, the compression quality is lossy given that it is a non-reversible operation. When involving quantisation, it can be immediately applied to data or within the encoding process of other compressors. For instance, quantisation is commonly utilised within other transform-based compressors such as DCT and DWT to obtain more significant compression. The process started by defining quantisation level L , which uniformly or non-uniformly subdivides the quantum into equal or non-equal regions, respectively, based on the quantisation type. In a stream, each value x will be quantised to the nearest member of the discrete measure belonging to a set of L . The range of any quantiser is limited by a x_{min} and x_{max} values. In a uniform quantisation, the steps have equal amplitudes, wherein a quantiser discards signal information lying between those levels. An example for computing equal regions can be defined using equation 2.7. A more mathematically complicated and more popular methodology is known as adaptive quantisation. Generally, the difference between the original analogue signal

and the quantised signal or the original input value and its quantised value is known as the quantised error or noise.

$$\Delta = \frac{x_{max} - x_{min}}{L} \quad (2.7)$$

2.3.3 Image Compression

Image compression is an essential and standard domain of science belonging to the data compression, signal processing, and computer vision fields. An image comprises several pixels, wherein colour and brightness values are stored at each pixel location. Thus, the image size is influenced by the number of pixels it contains and the bit size defined to store the pixel colour. Typically, a size of 8-bits per pixel is usually used to store 256 colours; however, more storage can be utilised up to 24-bits per pixel. Examples of file formats in which a digital image could be saved vary, including JPEGs, TIFFs, GIFs, and PNGs. Each format offers various usages while internally supporting different compression qualities. Thus, selecting which format to use usually depends on the application requirements' demand [115]. Regardless of the compression quality, given that image has a specific format, any compressor would generally gain a reduction in bit rate by utilising some knowledge and underlying structures. As an image naturally contains many types of redundancies (e.g. spatial, visual, and coding), by exploiting such repetitions, a coder gain reductions. Generally, given the structural similarity between images and videos, the applications of image codecs can further extend to be applied for video compressions. In the case of a lossy compressor, algorithms usually balance between maximising the achievement in reduction while minimums the impact on data fidelity. For instance, preserving the essential visual information while reducing the least important visual redundancies to allowable information loss. On the other hand, lossless compression gains an acceptable compression ratio with no loss of fidelity or quality, retrieving the original data. When comparing only the bit-rate amount, it is noticeable that the lossy methods produce a more significant reduction than a lossless method. However, such a feature comes with payback, causing higher chances for non-retrievable information loss.

2.3.4 Compression Performance Metrics

An essential step for any compression algorithm is to effectively measure the compression performance, especially when comparing outcomes to alternative state-of-the-art techniques. Within the compression literature, numerous standard metrics and measurements are commonly used. For instance, in the image compression domain, a measurement known as bits-per-pixel (bpp) is typically computed to measure the number of bits required to be stored per pixel as shown in equation 3.3.1. A similar metric concept applied in the text compression domain known as bits-per-character (bpc) that estimate the number of bits required to be stored per character. Other

data domains use a similar metric to evaluate the reduction performance known as Compression Ratio (CR), which is the proportion of the uncompressed to the compressed data size presented in equation 2.8.

$$CR = \frac{\text{Uncompressed Data Size (Bits)}}{\text{Compressed Data Size (Bits)}} \quad (2.8)$$

When the aim is to compare methods' compression speed, measurement of encoding and decoding time can be utilised. *Encoding time* is the entire time required to compress data, while *decoding time* is the total time needed to decompress data. To relatively measuring the space-saving between compressed and uncompressed data equation 2.9 can be used.

$$\text{Space_Saving} = 1 - \frac{\text{Compressed Data Size (Bits)}}{\text{Uncompressed Data Size (Bits)}} \quad (2.9)$$

Within deep learning literature, the comparison between codecs may consider the model's size, as the model's architecture can be massively large, consisting of numerous layers stacked in depth.

2.3.4.1 Medical Image Compression

Unlike digital photos, medical images are acquired and stored differently (see section 2.2 for more details). Medical images usually maintain a single channel representing intensity value per pixel/voxel. The stored intensity values may express a high feature space and data ranges based on the scanning setting and quality. Each scanning modality illustrates distinct body aspects or reflects specific functionality. Numerous image-based codecs are also employed in compressing medical images.

2.4 Deep Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI), which is one of the Computer Science sub-fields gaining significant attention and research contributions in the last decade. The scope of this sub-field includes designing and training algorithms to perform the mapping of data distributions and data estimations implicitly. The fundamental motivation is to simulate human intelligence in processing information and reasoning behaviour. Within the machine learning field, overlaps between various scientific areas are employed, including probability theory, statistics, approximation theory, convex analysis, and algorithm complexity theory. The applications of ML are thriving, including several domains, problem-solving tasks, and data types [116–119]. Example of fields where machine learning is utilised includes Computer Vision, Natural Language Processing (NLP), Bioinformatics, and Medical Diagnosis. Other examples of the state-of-the-art tasks are image classification, image segmentation, natural language processing, data compression, generative models that synthesise images, text, or audio data.

Deep Learning (DL) can practically be classified as a specialised subset of machine learning. DL generally designs artificial Neural Network (NN) that implicitly learn patterns, features, or distributions from given data and improve accuracy over time. These networks are composed of layers with some hidden variables adjusted during training. A model containing a few layers is known as shallow network architecture, while models with deeper (i.e. layers are stacked in-depth) and wider layers (i.e. the number of nodes per layer) are called Deep Neural Network (DNN) architecture. Generally, as the number of layers and the number of hidden variables increase, this improves the representation and learning capacity of the network. The network's capacity is essential as the amount of data grows; the models are usually expected to learn knowledge of the data, make predictions, or help in decision making based on the given problem while processing information efficiently. Such ability appears in many applications nowadays, like recommender systems, language translation, voice recognition, and real-time object detection [116, 120]. The DL involves not only the development of these architectures but also optimising them during training to solve both supervised and unsupervised learning processes.

2.4.1 Neural Networks

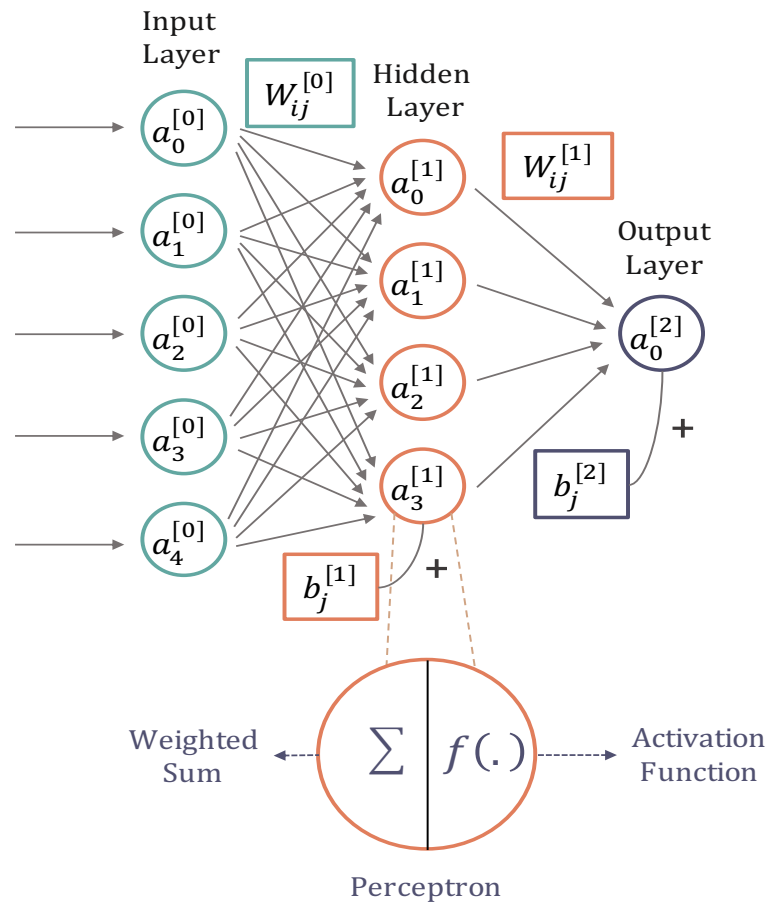


Figure 2.16: An example of a basic Multi-Layer Perceptron that composes of three layers types: input, hidden, and output, respectively. The first layer $a_i^{[0]} = x$ has five input features that form the raw data inputs, while the hidden layer $a_i^{[1]}$ contains four densely connected nodes, and the output layer $a_i^{[2]}$ has only one node. Each hidden connection is weighted by $W_{ij}^{[l]}$, each node (perceptron) is associated with bias scalar, and an activation function $f(\cdot)$ that are applied in equation 2.10.

Neural Network, also known as Artificial Neural Network (ANN), is formed by joined algorithms that simulate the human brain's neuron system. Most networks would contain three types of layers, specifically input, hidden, and output layer as shown in Fig. 2.16. When a network only consists of these three layers, it will be recognised as a shallow neural network, while complex and deeper architecture is distinguished as Deep Neural Network (DNN) architecture. A Multi-Layer Perceptron (MLP) is a class of feed-forward artificial neural networks, where each perceptron is computed

by equation 2.10 at the basic form. Exceptionally, the input layer is the only layer denoted as $a^{[0]} = x$, where $a^{[0]}$ is the input layer and x forms the raw input data or features. Each perceptron belonging to a hidden or output layer (e.g. $a_0^{[1]}$) is uniquely computed by the weighted sum of the perceptrons' outputs from the previous layer (e.g. $[a_0^{[0]}, a_1^{[0]}, a_2^{[0]}, a_3^{[0]}, a_4^{[0]}]$) in addition to a bias term (e.g. $b_0^{[1]}$). A nonlinear activation function $f(\cdot)$ is then generally applied (e.g. $f(z_0^{[1]})$) to the resulted output (e.g. $z_0^{[1]}$), which further allows producing complex representations. An overview illustration of a single perceptron, which composes the weighted sum and an activation function, is shown in Fig. 2.16.

$$z_j^{[l+1]} = b_j^{[l+1]} + \sum_{i=1}^{N^{[l]}} a_i^{[l]} W_{ij}^{[l]}, \quad a_j^{[l+1]} = f(z_j^{[l+1]}) \quad (2.10)$$

Each $[\cdot]$ in a superscript explicitly indicates the layer number while a subscript i or j identifies the node number within the layer, and $N^{[l]}$ is the number of nodes per layer. $z_j^{[l+1]}$ is the weighted sum of feeding the input features $a_i^{[l]}$ from the previous layer scaled by the weights $W_{ij}^{[l]}$ while $b_j^{[l+1]}$ forms a bias scalar associated with the current node. Lastly, $a_j^{[l+1]}$ forms the output of the current perceptron.

2.4.1.1 Vectorisation

$$Z^{[l+1]} = B^{[l+1]} + W^{[l]} A^{[l]}, \quad A^{[l+1]} = f(Z^{[l+1]}) \quad (2.11)$$

The computation in equation 2.10 can be modified to leverage parallelisation and utilise GPU capabilities by vectorising the process throughout the training samples, as shown in equation 2.11. In this equation, matrix to vector multiplication is applied, where W is a matrix of size $(N^{[l+1]}, N^{[l]})$ while B , Z and A are vectors with following sizes: $(N^{[l+1]}, 1)$, $(N^{[l+1]}, 1)$, and $(N^{[l]}, 1)$, respectively. Each row in W corresponds to a set of training samples for one node, while each column represents nodes per layer (i.e. hidden unit). The term “hidden” indicates that these nodes' values are not observed or known during training. Practically, each hidden node is computed during training based on the provided input features, while the dense connectivities and the nonlinearity of its mapping function result in complex relations to other nodes. Consequently, such sophisticated relations and mapping functions form the core mechanism of NN ability in learning and modelling complex data distributions.

2.4.1.2 Layers

A layer is an essential topological sub-structure of a neural network, where some nodes are aligned. The mathematical computations would vary based on the layer type, while the primary mechanism is formed by receiving inputs from the previous layer and passing outputs to the next. Additionally, a layer can be stacked in-depth to establish a deeper network structure. Some well-known neural network's layers, including dense layer, pooling layer, convolutional layer, deconvolutional layer, and recurrent layer, are displayed in Figure 2.17. The intention of providing these various options is to enhance the model's nonlinearity mapping ability, improve detecting high and low-level features and increase robustness against training problems (e.g. overfitting).

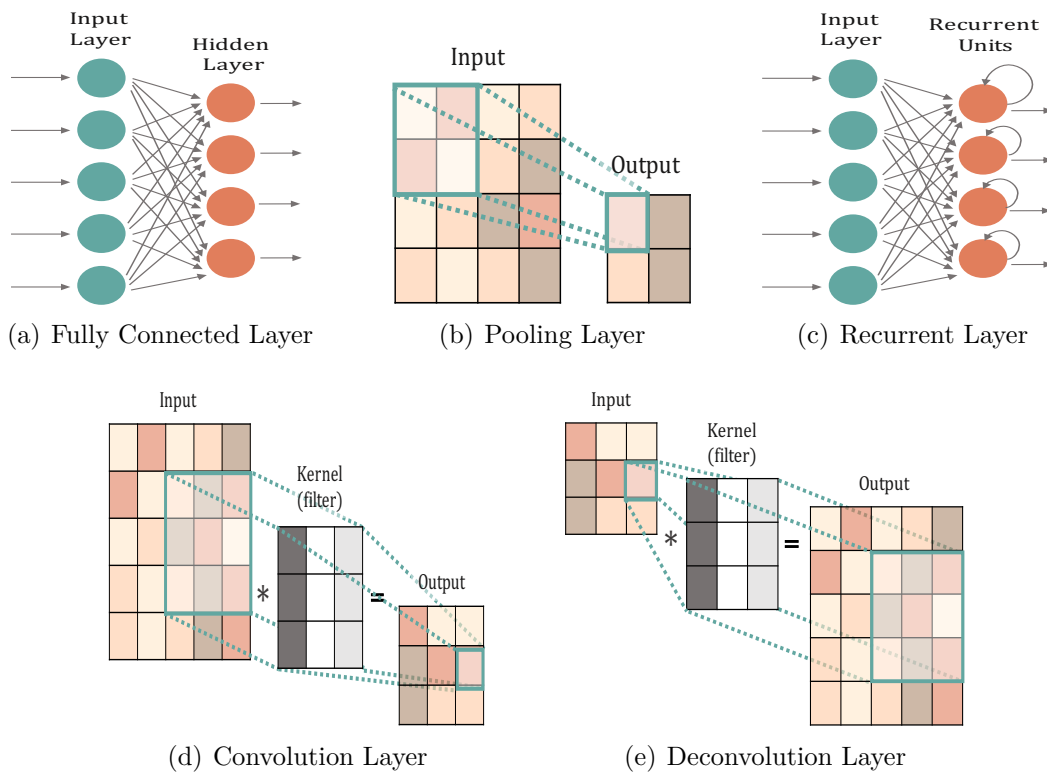


Figure 2.17: An illustration of some well-known neural network layers' types, including dense layer, pooling layer, recurrent layer, convolutional layer, and deconvolutional layer.

A dense layer is also known as a fully connected layer, where each node is topologically connected to all nodes in the preceding layer as illustrated in Fig. 2.17(a). On the other hand, a convolutional layer (ConvLayer) employs a mathematical operation called convolution, where a kernel (filter) is applied repeatedly over the input data,

as shown in Fig. 2.17(d). The convolutional layer has gained significant attention in numerous DL domains due to its ability to capture local patterns and extract structural redundancy within N-Dimensional signals, such as images and videos [35, 36]. The learned patterns typically started from simple low-level features like edges, surfaces, and curves up to more complicated structural shapes learned within deeper ConvLayers. The region covered by the kernel is known as the receptive field (i.e. determined by the kernel size), whereas element-wise multiplications are computed and summed. The output of the convolution layer is called “activation map” or “features map”. Usually, as the number of filters applied to the input information increases, the depth of the features maps grows, further expanding the knowledge about the input. While convolving the filter over the input data, two main parameters need to be assigned (i.e. stride and padding size). Practically, the stride controls the amount of shifting a kernel employs while sliding over data. In contrast, the padding defines the size for padding input’s border (e.g. padding input border with zero value). Therefore, padding is commonly applied to maintain the same dimensionality between the input and the resulting feature maps known as “Same Padding”. The dimensions (width and height) of the output activation map (O_h, O_w) are computed using the equation 2.12, where (I_h, I_w) are the input dimensions, P is the padding size, (F_h, F_w) are the filter sizes, and S is the stride amount.

$$O_h = \left\lfloor \frac{I_h \times 2P - F_h}{S} + 1 \right\rfloor, \quad O_w = \left\lfloor \frac{I_w \times 2P - F_w}{S} + 1 \right\rfloor \quad (2.12)$$

A deconvolution layer performs a revised mathematical operation of the convolution layer as shown in Fig. 2.17(e). Starting from the feature map output, a deconvolution operation would utilise filtering to retrieve the original input data. The dimensions of the resulted features (O_h, O_w) are calculated by Equation. 2.13.

$$O_h = S * (I_h - 1) + F_h - 2P, \quad O_w = S * (I_w - 1) + F_w - 2P \quad (2.13)$$

A ConvLayer is usually followed by a pooling layer (downsampling layer) to reduce the data dimensionality and simplify the computation complexity by providing abstract representations as illustrated in Figure 2.17(b). A pooling layer mainly decreases the computational power required to process the feature maps while effectively maintaining model training by retaining the most dominant features. Such extraction is performed by selecting a sub-sampling discretisation, namely average or maximum pooling (max pooling) options. While an average pooling summarises feature maps (compute the

average value for each patch), a max-pooling selects the most activated values within the input features (select the maximum value for each patch). Usually, the max-pooling performs better than the average one due to its ability in reducing noise by discarding noisy activations. The pooling layer follows the same principle as a convolution layer regarding a window-based application by assigning padding and stride parameters.

A recurrent layer is a type of NN layer that facilitates “looping” capability by processing not only input information but also output from previous layer calculations as illustrated in Fig. 2.17(c). Such ability allows remembering some knowledge of the data for a period of time. Most recurrent layer maintains internal memory or internal state to facilitate information during learning effectively, while some types facilitate a gating mechanism to control the amount of data to read, write, and store. By retaining the state information across successive inputs, recurrent layers obtain state-of-the-art results in many tasks that employ sequential data such as natural language and time series [120,121]. The recurrent cell can be classified into three main types: standard recurrent cell [122], Long Short-Term Memory (LSTM) unit [123], and Gated Recurrent Unit (GRU) [124]. Each of these types are covered in detail in sub-section 2.4.2.

2.4.1.3 Activation Functions

An activation function $f(\cdot)$ is a core in adding differentiable nonlinear mapping that allows encoding complex input patterns and controls how well a model learns during the training process. Several well-known nonlinear activation functions can be applied, including Sigmoid (σ), Hyperbolic Tangent (Tanh), Rectified Linear Units (ReLU) [125], Leaky-Rectified Linear Unit (LeakyReLU) [126], and Exponential Linear Unit (Elu). A selection of some activation functions commonly applied in neural networks is presented in table 2.3. As each function has various features, the activation choice results in a diverse capacity for the data modelling. While an individual layer holds the same activation function, various activations may be involved within different parts of the same model. For instance, the hidden layers may have the same activations, while a different activation can be applied to the output layer to control the output type or the prediction type based on the task requirements.

A sigmoid activation function $f_{\sigma}(z)$ is a well-known logistic function that returns a value range of $[0, 1]$. The sigmoid function generates a nonlinear result with S-shape curve plot, which can be computed using equation 2.14. When the input z is a considerable positive value, the result of applying Sigmoid is close to 1, while a large negative input would result in an output close to 0. Such output range would be

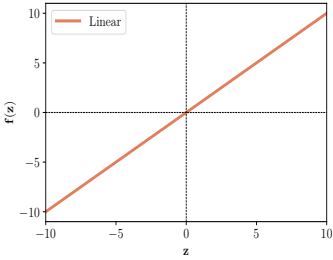
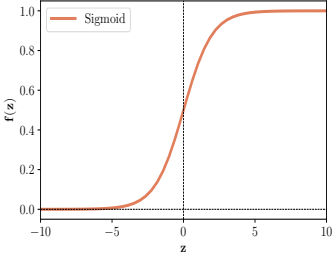
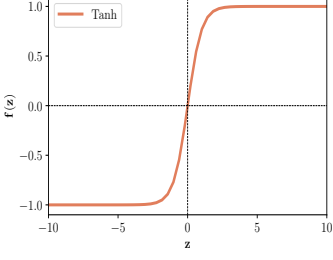
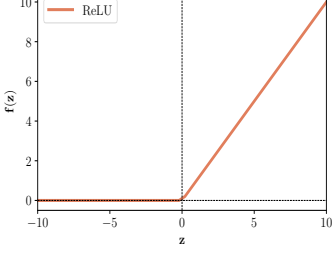
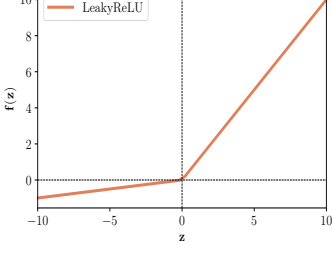
Name	Plot	Equation	Range
Linear, or Identity		$f_{linear}(z) = z$	$(-\infty, \infty)$
Sigmoid, or Logistic (σ)		$f_{\sigma}(z) = \frac{1.0}{1.0+e^{-z}}$	$[0, 1]$
Hyperbolic Tangent (Tanh)		$f_{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$[-1, 1]$
Rectified Linear Units (ReLU) [125]		$f_{ReLU}(z) = \max(0, z)$	$[0, \infty)$
Leaky-Rectified Linear Unit (LeakyReLU) [126]		$f_{LReLU}(z) = \max(\alpha z, z)$	$(-\infty, \infty)$

Table 2.3: A selection of activation functions commonly applied in neural networks.

problematic when multiple hidden layers have Sigmoid activations, causing the training to be slow due to gradients' vanishing. When the input value z is small or big, the gradient is zero, the curve of this function is barely changing, which cause the learning

process to be negligible and inefficient. However, a sigmoid activation is commonly applied in an output layer for binary classification problems.

$$f_{\sigma}(z) = \frac{1.0}{1.0 + e^{-z}} \quad (2.14)$$

The Hyperbolic Tangent (Tanh) activation function $f_{Tanh}(z)$ (equation 2.15) has similar S-shape curve function as $f_{\sigma}(\cdot)$ while the range expanded to be in the range $[-1, 1]$. When the input z is a large positive value, the result of applying Tanh is close to 1, while a large negative input would result in an output close to -1 (a zero centralised function). Although this function allows output features to carry negative values to the next layer, it is also susceptible to vanishing gradient problems, but it is still preferred over Sigmoid activation.

$$f_{tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.15)$$

The Rectified Linear Units (ReLU) function $f_{ReLU}(\cdot)$ is perhaps the most common nonlinear function used for hidden layers due to its effectiveness and less susceptibility to gradients vanishing [125]. The ReLU function as presented in (equation 2.16), gives a linear output of z when the input z is positive; otherwise, the output will be 0 (value range: $[0, \infty)$). Compared to other activation functions, namely Sigmoid and Tanh, ReLU is computationally less expensive, requiring uncomplicated mathematical operations. However, one disadvantage of the ReLU function is that it would result in some “dead” units or neurons when the activation is negative, causing some gradients to be zero during training, which restricts the model from fully utilising its capacity.

$$f_{ReLU}(z) = \max(0, z), \quad f_{ReLU}(z) = \begin{cases} z & z > 0. \\ 0 & \text{Otherwise.} \end{cases} \quad (2.16)$$

To overcome the “dead” unit problem, the Leaky-Rectified Linear Unit (LeakyReLU) activation function $f_{LReLU}(\cdot)$ can be applied, where a slight positive slope in the range $[0, 1]$ is involved [126]. Such negligible slope allows keeping the gradient updates alive. The equation of the LeakyRelu is a variant of ReLU, instead of 0 activations when $z \leq 0$, a small non-zero constant gradient α is scaling the input

z , as shown in 2.17. Commonly, the constant value of $\alpha = 0.01$ is assigned.

$$f_{LReLU}(z) = \max(\alpha z, z), \quad f_{LReLU}(z) = \begin{cases} z & z > 0. \\ \alpha z & z \leq 0. \end{cases} \quad (2.17)$$

A Linear activation function $f_{linear}(z)$, as the name suggests, it is a function that implies linear mapping; it is simply proportional directly to the input as shown in equation 2.18. This function can be referred to by many names such as “identity” or “no activation”. The usage of this activation is perhaps reluctant to the output layer as it allows mapping the output of NN to an unlimited range of values, possibly when solving linear regression problems.

$$f_{linear}(z) = z \quad (2.18)$$

2.4.1.4 Loss Functions

The terms loss function, cost function, or error function refer to the same concept, which is generally an optimization method used to assist the learning process of a NN model. A cost function’s main objective is to evaluate how well a model performs on the entire given data, while a loss function evaluates how well a model performs on a single training example. Estimating the error amount is an essential step in the learning process as it helps reduce the loss in the following evaluation by updating models’ weights in the right direction. The more precise the selection of the loss function is, the more efficient the error estimation becomes. Based on the problem domain and task requirements, the choice of cost function may vary. Regardless of the type, the optimization strategies commonly aim is to minimize the cost function. Broadly, in deep learning, the supervised problem domain can be specified into some types, such as classification or regression. When solving a regression problem, regression losses can be involved, unlike classification losses, which should be defined to evaluate classification problems. A brief description of the differences between the two problem types is: classification output is usually a set of finite categorical values (e.g. classes, categories and labels), whereas regression deals with predicting continuous values (e.g. quantities, amount, and sizes) – further details will be covered in sub-section 2.4.3.

As classification tasks typically require predicting the probability of a discrete class or set of classes for the given raw input values, the defined loss usually measure the difference between probability distribution functions (e.g. the predicted output’s labels and the ground truth labels). Therefore, when defining the model’s output vector,

the number of output labels should match each possible class. Then, by applying a Softmax activation function or Logistic function on the resulting “logits”, a greedy decision boundary between these features will specify the classification choice that has the most significant activation (high certainty). All the output values of this function will sum up to one (a probability value in the range $[0, 1]$) as presented in equation 2.19. Examples of classification losses include Hinge loss/Multi-class SVM loss, Cross-Entropy Loss/Negative Log-Likelihood loss, and Kullback-Leibler Divergence Loss.

$$\hat{y}_i = \frac{e^{a_i}}{\sum_{j=1}^C e^{a_j}} \quad (2.19)$$

On the other hand, within the regression problem domain, a model is trained to predict a specific value (i.e. real-valued quantity) that is continuous in nature. Therefore, the defined loss function typically focuses on computing the residual or the differences (distance-based) between the predictions and the original target values. Broadly, based on the cost function score, the lower the value, the higher the model’s performance. The objective functions reach the minimum when the prediction is precisely equal to the actual value (i.e. the residual is zero). Empirically uncomplicated but robust well-known regression losses are the Mean Squared Error (MSE) Loss, Mean Absolute Error (MAE) Loss, and Huber Loss.

The Mean Squared Error Loss (MSE), also known as L_2 loss, directly computes the average of the squared difference between the ground truth and the predicted values as presented in equation 2.20. The main drawback of L_2 loss is its susceptibility to outliers as it gives a significant penalty for outliers (due to the quadratic form), which reduces its robustness. When the model’s error is significant, this affects the stability of the training due to the extremely steep gradient, which may lead to an “exploding gradient” problem.

$$L_2 = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (2.20)$$

The Mean Absolute Error Loss (MAE), L_1 loss, and Laplace loss refer to the same objective function, which aims to minimize the average sum of all the absolute residuals as calculated in equation 2.21. Unlike the L_2 loss, the MAE loss has less sensitivity to outliers when the error value is significant, while it practically struggles with small error values as its derivative is discontinuous. When the error has a slight variance among observations, the L_1 loss results in the same gradient throughout,

making the gradient significant even for small loss values, which causes instability in learning and may lead to missing the minima.

$$L_1 = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (2.21)$$

The Huber loss also called the Smooth Mean Absolute Error loss, has the favourable characteristics from both L_1 and L_2 losses. The Huber loss is robust against outliers and differentiable at the minima (i.e. zero residual). When the error is slight, Huber loss becomes quadratic similarly to MSE, while significant errors are handled linearly like MAE. The computation choice is tuned based on a hyperparameter known as δ (delta) as shown in equation 2.22. The only challenge for this loss is the selection of the delta, which may require iterative training.

$$L_\delta = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, & \text{if } |y_i - \hat{y}_i| \leq \delta. \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases} \quad (2.22)$$

2.4.1.5 Training by Back-Propagation

Any neural network is trained using a mechanism called Back-Propagation Through Time (BPTT). Generally, the model’s training is formulated to find network attributes (e.g. W_* and B_*) at which the cost function $C(\cdot)$ achieves a minimum value (minima). In the beginning, all the model’s parameters are initialised. The learning has a forward pass, where the training input patterns (batches) propagate through the network, outputting some predictions. Then, a loss function is involved in computing the error by comparing the predictions to the target outputs. Next, the optimiser calculates the derivatives with respect to the current set of weights and adjusts the network attributes to minimise the error. Iteratively, applying forward and backwards passes for a specific number of epochs or training steps while updating the network attributes and reducing the cost function.

A “batch” forms a subset of the training dataset, where couples input-output training samples are combined. A single pass through all samples belongs to the training dataset while updating the network’s weights is commonly called “Epoch”. Thus, it could be inferred that a single epoch is composed of one or more training batches. During Back-Propagation, the optimizer computes the gradient of the loss function with respect to each weight by the chain rule.

2.4.1.6 Optimizer

The core of the neural network learning process is the optimisation algorithm, also known as an optimiser, which guides the overall training and improvement of a model while solving the task at hand. The term optimisation usually refers to mathematically minimising (or maximising) any expression or function. As the cost function C measures a model's error score on a given data, the optimiser plays an essential role in improving the model by updating the model's parameters to minimise the error rate. The learning process combines many hyperparameters to tune and adjust, including the model's weights, biases, and learning rate. These hyperparameters are iteratively tuned during the training by the optimisation algorithm to get an optimal solution (e.g. global optimum or minima) that potentially best fit the given observed samples. Numerous optimisation algorithms are available with different properties and performance specifications such as memory requirements, processing speed, and numerical precision. Three of the most commonly utilised optimisation algorithms are Gradient Descent (GD) [127], Stochastic Gradient Descent (SGD) [127], and Adaptive Moment Estimation (Adam) [128].

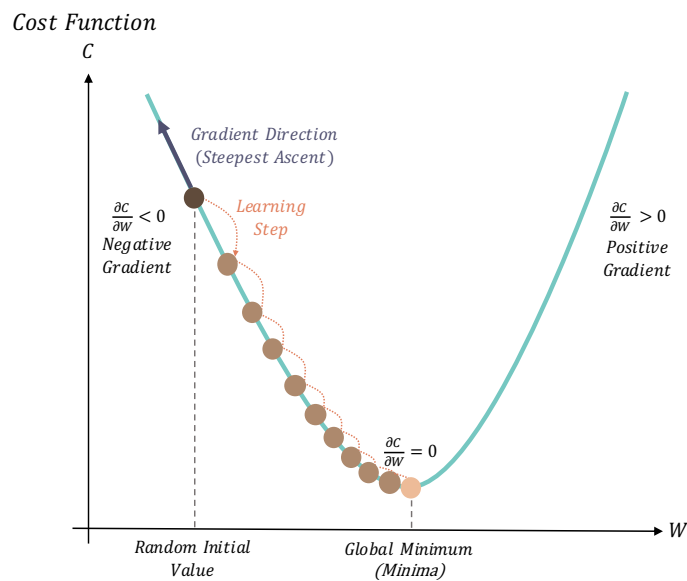


Figure 2.18: An illustration of a gradient-based optimisation algorithm that minimises an objective function C by moving in the opposite direction (downhill) of weight's gradient $\frac{\partial C}{\partial W}$.

Computing the derivative of the cost function C with respect to each network attribute is mandatory to tweak all the weights and biases values for the given input

and solve the task at hand. Basically, finding the derivative is essential to determine the direction of the steepest descent (the gradient’s opposite direction). A gradient is the “slope” of a function at a given point while its direction points directly uphill. Thus, to minimise the cost function C at each iteration, the current weights and biases need to be adjusted by their negative gradients as presented in equation 2.23 and 2.24. Where W_{ij} and B_{ij} denotes the current weight and bias values while $\frac{\partial C}{\partial W_{ij}}$ and $\frac{\partial C}{\partial B_{ij}}$ forms the partial derivatives of the loss function with respect to the weight and bias. Each partial derivative indicates the change to be made to optimise a network’s parameter (e.g. weight and bias). α is the learning rate hyperparameter, which controls the amount of learning (the learning step) that will be taken at each iteration. The symbol $:=$ denotes an updating process to the current parameter value. The optimisation algorithm iteratively tunes the network’s parameters at each layer in the direction by which the objective function C is decreasing. An illustration of a gradient-based optimisation algorithm, also known as gradient descent algorithm, which minimises an objective function C by moving in the opposite direction (downhill) of the weight derivative as presented in Fig. 2.18.

$$W_{ij} := W_{ij} - \alpha \frac{\partial C}{\partial W_{ij}} \quad (2.23)$$

$$B_{ij} := B_{ij} - \alpha \frac{\partial C}{\partial B_{ij}} \quad (2.24)$$

Gradient Descent (GD), also called Batch Gradient Descent, is a standard optimisation algorithm that minimises an objective function by iteratively taking steps in the direction of negative gradients until reaching the correct model’s parameters [116, 127]. The main drawback of the gradient descent method is that it can readily get stuck in a local minimum due to its massive reliance on the initialisation of the parameters. Another drawback is related to the parameters adjustment, as GD allows the model to update after propagating through the entire dataset in a single iteration, which constrains the implementation by the memory capacity, causing the gradient convergence to be slow and naturally creating a significant gradient value. Such updating mechanism would generate massive updating steps that may overshoot the optimal value without reaching it. Stochastic Gradient Descent (SGD) differs from the GD algorithm as it updates gradient’s immediately after each sample [127]. Such methodology allows faster convergence than the GD even on a large dataset. However, it may cause noisy jumps in the training steps as the performance on each training example influences each gradient update. An enhanced version is the Mini-batch Gradient Descent, where

mini-batches (training instances) with multiple training samples are fed to the model with no replacement while updating the gradient after each instance [129]. Such an optimisation algorithm allows less noisy training steps as it reduces the variance in the parameters update with faster convergence and higher stability. A reduced smooth oscillation of learning steps yet accelerated convergence can be achieved by adding a momentum term, whereas the variation in the model's parameters intends to be in the same direction when training samples follow similar patterns. By adding the momentum term, the optimiser may be known as Stochastic Gradient Descent with Momentum (SGDM), which changes the parameters update to be the exponentially weighted average (moving average) of the gradients instead of only the gradients values. Another popular and complicated optimisation algorithm is the Adaptive Moment Estimation (Adam), which also utilises the concept of momentum as well as involving adaptive learning rate [128]. Adam optimiser introduces better training stability as the future gradients updates are determined linearly by the gradients' updates from previous iterations using an exponentially decaying mean gradient (first moment) and variance (second moment). In Adam optimiser, each network parameter is adjusted using an individual adaptive learning rate, which allows estimating the gradients' first and second moments as the training progresses. Such adaptation of learning rates leads to empirically better convergence without the need to manually tune hyperparameters.

2.4.2 Recurrent Neural Networks (RNN)

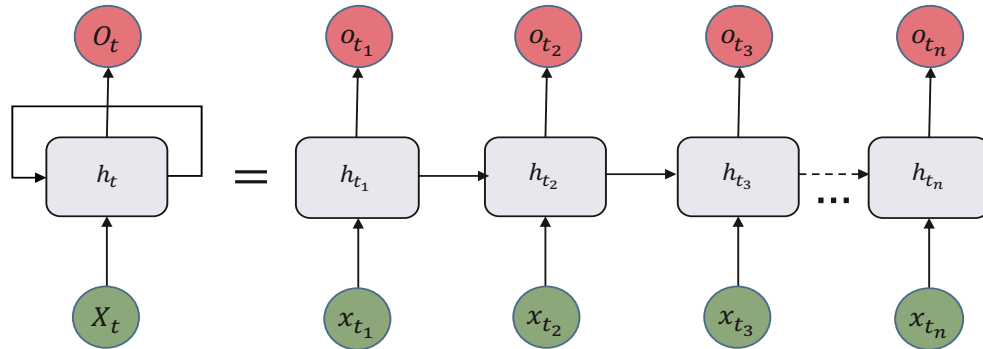


Figure 2.19: Demonstration of a standard Recurrent Neural Network (RNN), which consists of multiple input observations (X_t), memory/internal states (h_t), and multiple output values (O_t). The illustration (on the right) is an example of an unfolded recurrent neural network.

Recurrent Neural Network (RNN) is a particular type of neural network architecture, where recurrence or loop connections are involved within the architecture to feed the network activations from a previous time step [122]. Moreover, one of its main features is maintaining a hidden internal state that allows storing and memorising information through time stamps. Such design allows RNN to learn a mapping from a prior input sequence of observations to the desired output stream while capturing contextual dependencies. Demonstration of a standard RNN architecture is shown in Figure 2.19, while a detailed illustration of operations within the RNN cell is provided in Figure 2.20(a). Compared to the MLPs, the RNN model can be considered a more generalised version, maintaining input and output series of arbitrary length. As the RNN strategy allows looping over the input sequence, it is naturally capable of solving various tasks, including sequence prediction problems, time-series forecasting problems, text processing, speech recognition, sequence analysis, and machine translation tasks. However, similar to a deeply NN architecture, a deep RNN model suffers from training instability with a higher chance of facing gradient vanishing or explosion problems. This issue is because the same weights are used for each time-step, while only the outputs and the internal states differ. To avoid such problems, variations of RNNs were proposed (i.e. Long Short Term Memory Units (LSTMs) and Gated Recurrent Units (GRUs)) with better training stability than the traditional RNN.

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \quad (2.25)$$

$$y_t = \sigma_y(W_y h_t + b_y) \quad (2.26)$$

Where x_t denotes the input vector, h_t is the hidden layer vector, y_t is the output vector, W U_h b are weight matrices and vector, and σ_h σ_y represent activation functions.

2.4.2.1 Long–Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) is one of the most widely used classes of RNNs [120, 123, 130]. LSTM main strength is its 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 stabilised 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 order set of observations that passes sequentially through the hidden cells of LSTM. During training, LSTM adaptively modulates the flow of information using the gates while capturing dependencies of different time steps.

Within each cell, three main gates (input (2.28), output (2.29), and forget gates (2.27) equations) are handling the amount of information that is processed, omitted or stored. Each cell maintains an internal state or a memory unit (2.31) that updates while aggregating the input sequence through time before making a prediction. Moreover, Sigmoid and Tanh activation functions are employed within the cell to add some nonlinearity representation, whereas a zero value stops the data flow, and one allows it. Each gate handles the flow of information differently. To be precise, the input gate controls whether the memory unit is updated, while the forget gate chooses whether the memory unit is reset, and finally, the output gate determines whether the state of the current cell state is made visible. A detailed overview of LSTM cells' gates is provided in Figure 2.20(c).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (2.27)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2.28)$$

$$y_t = \sigma(W_y[h_{t-1}, x_t] + b_y) \quad (2.29)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad (2.30)$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \quad (2.31)$$

$$h_t = y_t \otimes \tanh(C_t) \quad (2.32)$$

Where x_t is the input sequence, h_t is the current cell output, C_t is the current cell memory, \tilde{C}_t is the candidate for cell state at timestamp t , while h_{t-1} is the previous cell output, C_{t-1} is the previous cell memory, and y_t is the output vector. W and b are learnable parameters (weights and biases), \otimes denotes element-wise multiplication, and σ is the Sigmoid activation function.

2.4.2.2 Gated Recurrent Unit (GRU)

A Gated Recurrent Unit (GRU) is another popular variation of RNN, utilising some gating mechanisms [124]. Although fewer gates are employed in the GRU cell, it still retains better training stability in handling vanishing gradient problems than traditional RNNs. Unlike LSTM, GRU contains two gates, and only the hidden state transfers the information without the cell state. The gates in GRU are composed of a reset gate and an update gate. Similarly to the LSTM’s forget gate, an update gate determines whether the information will be streamed or deleted. The other gate is the reset gate, which controls the amount of past information to forget. Such gating schemes mitigate short-term memories with adaptive filtering, whereas some sequences’ values may be kept in memory for a certain period while influencing data at future time stamps. Compared to the LSTM cell, GRU training can be slightly faster due to the fewer computations. A detailed overview of GRU cells’ gates is provided in Figure 2.20(b).

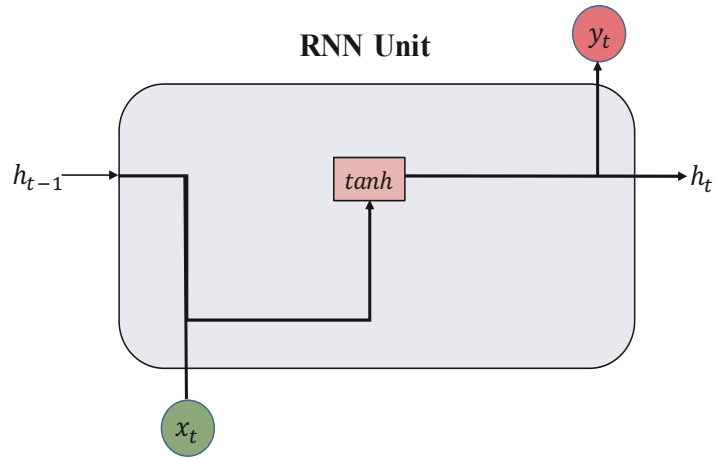
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (2.33)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad (2.34)$$

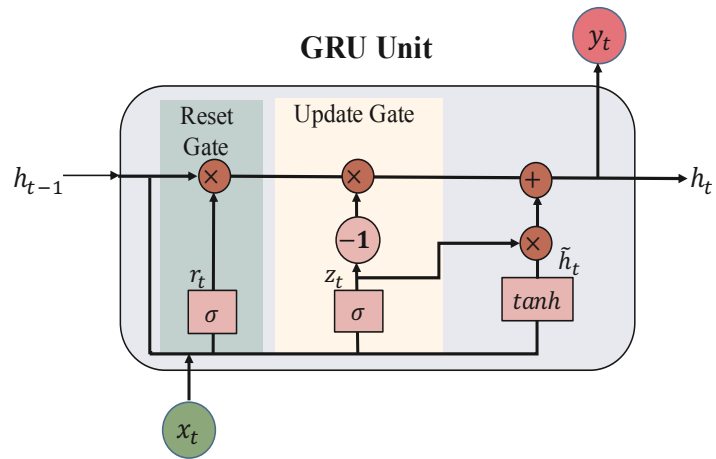
$$\tilde{h}_t = \tanh(W_h \cdot [r_t \otimes h_{t-1}, x_t] + b_h) \quad (2.35)$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t \quad (2.36)$$

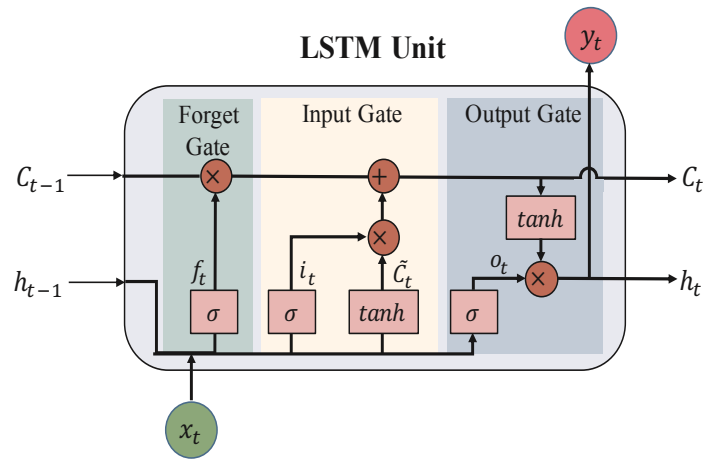
Where x_t is the input sequence, z_t denotes the update gate at time step t , r_t denotes the reset gate, h_t is the current cell output, \tilde{h}_t is the candidate for cell hidden state at timestamp t , and h_{t-1} is the previous cell output. W and b are learnable parameters (weights and biases), \otimes denotes element-wise multiplication, and σ is the Sigmoid activation function.



(a) RNN Unit



(b) GRU Unit



(c) LSTM Unit

Figure 2.20: An illustration of the different RNN cell classes, including traditional RNN unit 2.20(a), GRU unit 2.20(b), and LSTM unit 2.20(c).

2.4.3 Supervised Machine Learning

Generally, within the machine learning field, the task to be solved can be categorised as supervised or unsupervised learning tasks. Briefly, unsupervised learning is defined when a model is trained on an unlabelled dataset, implying that only input samples are available without output labels. In such problems, given raw unlabelled data, the learning-based model usually attempts to find representative forms of the data in lower dimensionality (e.g. clustering, dimensionality reductions, or data embeddings). On the other hand, a problem is described as a supervised learning process, when a matching output label is available for each input data point belonging to the training dataset (i.e. input-output paired samples).

2.4.3.1 Sequence Prediction Models

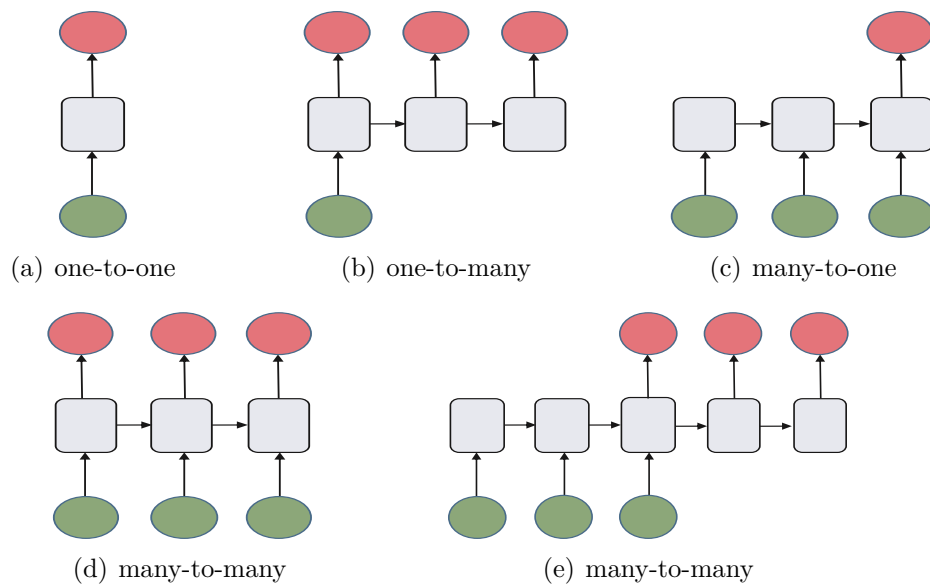


Figure 2.21: An illustration of the numerous learning-based sequence prediction models classified by the input-output sequences' length.

Sequence prediction problems are a particular type of supervised learning task whereby a model approximates a differentiable mapping function from input space to desirable output space. Within such a prediction-based problem, the input-output samples can be defined in arbitrary lengths, representing continuous or discrete ranges. This prediction-based model learns to capture the nonlinear patterns in the sequential input data over time while predicting the future values. The predicted output can represent the next value, class label, or sequence, which generally belong to one of

these classes: sequence prediction, sequence generation, sequence classification, or sequence-to-sequence prediction. A sequence is practically defined as an order set of prior observations that pass sequentially through a sequence prediction model before yielding target output result [120]. Based on the sequence specifications, models can be classified into various types. For instance, an one-to-one prediction model $\hat{f}(\cdot)$ yields a single output $\hat{y}(t)$ value for each input value $X(t)$ shown in Fig. 2.21(a) (e.g. a model predicting the next number). An illustration of the numerous learning-based sequence prediction models classified by the input-output sequences' length is presented in Fig. 2.21. Each prediction model has a broad range of domain applications covering numerous sequential and higher-dimensional data, gaining considerable attention within the deep learning research communities.

Formally, let $X = \{x_1, x_2, x_3, \dots, x_n\}$ be a finite set of input sequences defined over an input space \mathcal{X} , where $\mathcal{X} \subset R^d$, and $Y = \{y_1, y_2, y_3, \dots, y_n\}$ is the set of output sequences defined over an output space \mathcal{Y} , where $\mathcal{Y} \subset R^d$, and d denotes the dimensions. Let θ is called the parameters space, and θ_* is a specific parameter vector. A sequence prediction model \hat{f} is expected to learn a differentiable mapping function $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$ while adjusting the model's parameters θ_* through the back-propagation process given a training dataset \mathcal{D} . Generally, the process of finding such an optimal parameter vector θ_* is called training. The training dataset \mathcal{D} consists of finite input-output paired of samples defined as $\mathcal{D} = \{x_i, y_i\}_{i=0}^n$, where n is the total number of training samples. Each x_i represents an input vector $x_i = (x_t, x_{t+1}, \dots, x_{l_i})$ with $1 \leq t \leq l_i$ and l_i denotes the sequence length for all the input vectors, whereby $l_i \in N$. Similarly, each y_i represents the corresponding i-th label output vector $y_i = (y_t, y_{t+1}, \dots, y_{l_o})$ with $1 \leq t \leq l_o$ and l_o denotes the sequence length for all the output vectors, whereby $l_o \in N$. During training, a minimization of the loss function $\mathcal{L}(\hat{y}_i, y_i)$ is sought to reduce a specific distance measure between model's predictions \hat{y}_i and ground truth values y_i while updating the model's parameters θ_* .

An example of a one-to-many prediction problem would be image caption generation, where, for a single image, multiple words are generated illustration is in Fig. 2.21(b). Predicting weather forecasting, stock market, or product recommendation are well-known examples of many-to-one sequence prediction problems Fig. 2.21(c), wherein multiple prior sequence values are fed to the model while generating a single future value. Based on the number of future output values required to be produced, these problems can also naturally fit into a many-to-many category. DNA sequence classification and sentiment analysis are other many-to-one models belonging to sequence classification problems, where an output class is produced given a set

of input values. Many-to-many prediction models usually solve sequence generation problems such as text, music, handwriting, or image caption generation as illustrated in Fig. 2.21(d). A Sequence-to-Sequence (seq2seq) learning is a particular prediction models that solve problems with arbitrary input-output lengths, for instance, text summarization and multi-step time series forecasting as presented in figures 2.21(e).

As the input-output sequence can have a variable length over time, the architecture of a sequence prediction model should provide flexibility to maintain the dependency within such contextual information. Generally, many well-known sequence prediction models contain internal state or memory units that allow capturing and storing temporal dependency for some duration. The functionality and computation of this internal state vary from neural network architecture to another —(e.g. the difference in computations between LSTM and GRU cells see section 2.4.2 for more details). However, it is commonly used to retain previous information while influencing future output results. Examples of some state-of-the-art deep learning sequence prediction models are Recurrent Neural Network (RNN) [122], Long Short-Term Memory (LSTM) [123], Gated Recurrent Units (GRU) [124], and Transformers [121, 131, 132].

2.4.4 Challenges and Observations

- One of the intentions of utilising deep learning methods is to practically solve the assigned tasks by learning nonlinear representations or functions based on the available knowledge while lowering explicit feature crafting and manual decision making. However, such an objective is influenced by many non-trivial challenges such as selection of model architecture, training hyperparameters, the choice of loss function and optimiser.

2.4.5 State-of-the-Art Image Compression Techniques

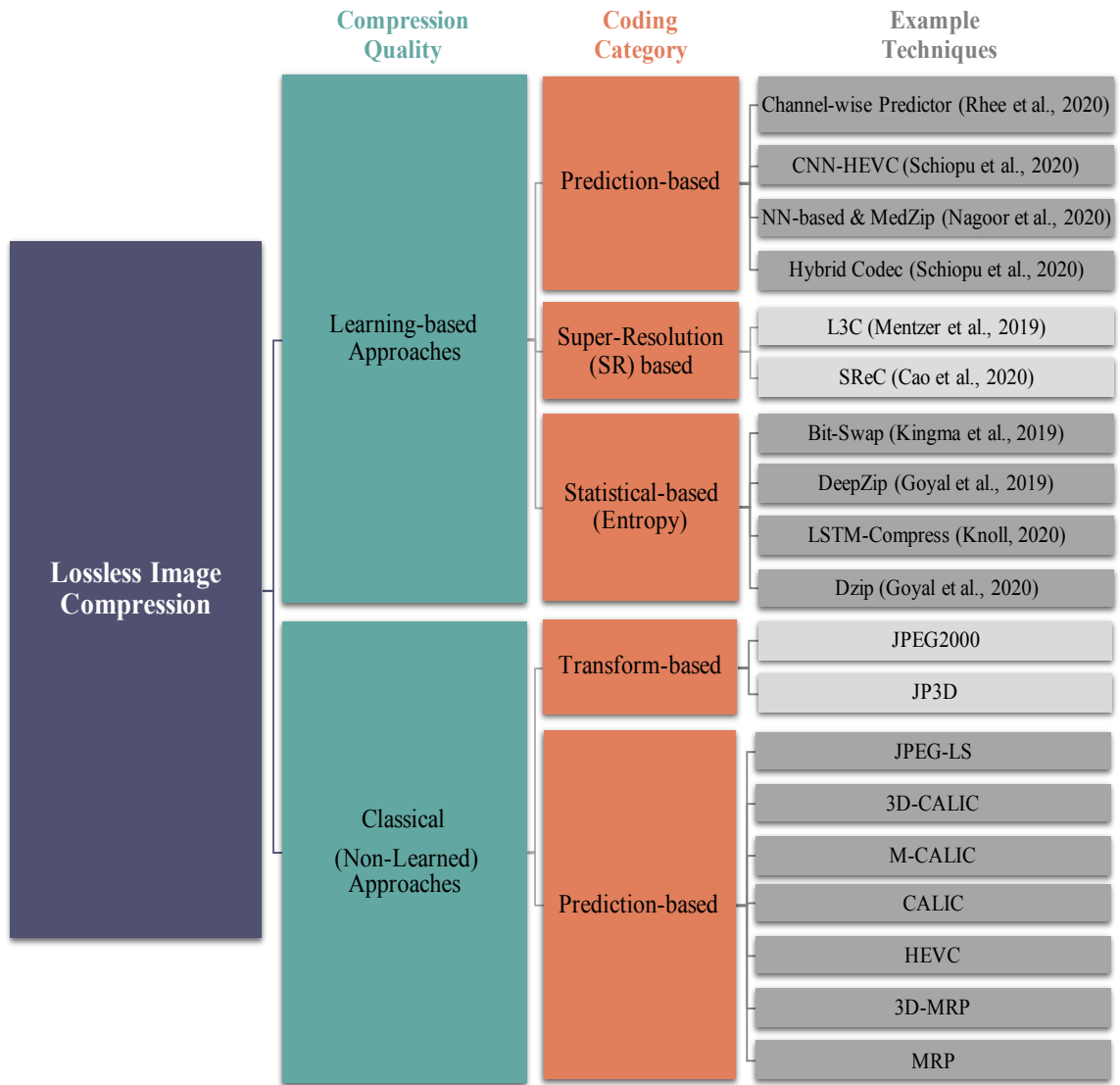


Figure 2.22: Highlighting some current lossless codecs literature, including classical (non-learned) and deep learning-based methods.

The level of compression that can be achieved depends on the type of patterns and assumptions that can be made about the target data, such as spatial, coding and spectral (psycho-visual) redundancies [133]. This section highlights the current lossless codecs trends, including classical (non-learned) compression literature and deep learning-based compression literature. An overall illustration of lossless image codecs is presented in figure 2.22. According to current compression research trends, image reduction methodologies can be classified into five main types: prediction-

based [9, 10, 13, 134–141], transform-based [7, 142], Super-Resolution (SR) based [4, 32], statistical quantisation/coding-based [3, 143–146], and end-to-end compression frameworks [147].

An alternative classification would be according to the dimensionality by which a coder computationally utilises data. A coder is categorised into a 1D codec when only sequential data (e.g. data from 1D space) is usually involved, which is generally the scheme of statistical/entropy-based compressors (e.g. Bit-Swap and DeepZip). On the other hand, if a codec is computed in 2D space (i.e. image), it is classified as an image encoder such as L3C, SReC, JPEG2000, CALIC, JPEG-LS, and MRP. Lastly, coders use information from higher dimensional space (e.g. 3D space) are classified into volumetric encoders such as HEVC, M-CALIC, 3D-CALIC, 3D-MRP, JP3D, MedZip, and CNN-HEVC.

2.4.5.1 End-to-End Compression Frameworks (Lossy)

Some contributions within the medical image compression domain but for lossy reduction are briefly specified in this subsection. A neural network model is defined to be an end-to-end compression framework when it follows a variational auto-encoder model, which is composed of trainable encoder-decoder architectures, yielding a compact latent representation. In such a model, the proposed reduced latent representation effectively captures the spatial information. Moreover, the model’s architecture can flexibly be composed of deep neural network types such as MLP-based, RNN-based, and CNN-based. An end-to-end lossy compression method using encoder-decoder RNN-based architecture combined with an entropy coding was presented by Toderici et al. [148]. Sushmit et al. [149] provide another learnable encoder-decoder based on convolutional Recurrent Neural Networks (RNN-Conv) specifically for chest X-ray images. A recent comprehensive study of end-to-end learning-based compression frameworks was proposed by [147]. As the study only focuses on solving supervised problems with lossy compression quality, considerable distortion metrics to measure and identify the perceptual artifacts yielded are involved. Such evaluation metrics include Peak Signal-to-Noise Ratio (PSNR) and Multi Scale Structural Similarity (MS-SSIM) [83, 150].

2.4.5.2 Entropy-based Compression Methods

A learning-based entropy/coding-based compression method usually combines a likelihood-based model with an entropy coder. The statistical modelling unit in-

tends to capture the underlying distribution of the input data through a NN-based model. On the other hand, the entropy coding unit involves an arithmetic coder or other Asymmetric Numeral Systems (ANS) that reduces the bit rate losslessly while ensuring identical reconstruction. Recent examples of learning-based entropy codecs are Bit-Swap [144], DeepZip [145], LSTM-Compress [146], and DZip [3].

Bit-Swap [144] contains hierarchical latent variable models with a Markov chain structure for compressing data losslessly while keeping bits-back coding efficiently compatible. The hierarchical latent variable forms a robust density estimator while combining it with ANS coding, compressing high-dimensional datasets efficiently. Empirically demonstrate Bit-Swap’s compression performance over well-known image-datasets such as MNIST, CIFAR-10 and ImageNet.

DeepZip [145] provides a lossless encoder-decoder framework wherein each model consists of two primary blocks: an RNN-based probability estimator and arithmetic coding. The probability estimator general architecture comprises RNN block (i.e. LSTM/GRU-based models), dense layer, and a subsequent softmax layer. The predictor maintains a many-to-many input-output mapping in which multi-input values are used while yielding multi-output values. An input sequence is divided into overlapping segments of length $K + 1$ (shifted by one), where K represents the previously encountered symbols fed to the predictor model. In DeepZip, the value of K was typically chosen to be 64. The generated output values will be merged before inputting them to the dense layer. The outcome of the model’s softmax layer is a probability distribution, which then will be compressed by an arithmetic coder. Based on the conditional probability of the next element in the distribution, this entropy coder uniquely determines a range per symbol belonging to a stream. When no significant improvement is noticed during training, early termination will be applied at a maximum of 10 epochs as an optimisation step. It is essential to know that after training, the model’s weights are stored as part of the compressed representation.

Like DeepZip, **LSTM-Compress** [146] also utilises LSTM predictors as probability estimators to adaptively learn the source distribution while condensing the learned representations into more compact forms with an arithmetic coding unit.

An enhanced and general-purpose lossless compressor uses NN-based modelling combined with arithmetic coding known as **DZip** [3]. This learning-based model can compress not only sequential data but also a variety of real datasets (regardless of the alphabet size), including text, genomics, executable files, audio data, and double-precision floating-point datasets. Similar to DeepZip, DZip maintains NN-based statistical data modelling combined with arithmetic coding. However, during training,

DZip utilises a hybrid training scheme at which a combination of semi-adaptive and adaptive training approaches are applied to two models, a bootstrap model and a supporter model, as illustrated in Fig. 2.23. The bootstrap intends to learn from the given data, while after training, the model’s weights are stored as part of the compressed representation. When estimating density, predicting the current symbol within a sequence is based conditionally on the probability of its past K symbols (with $K = 64$ being a default value). The prediction involves contributions from both bootstrap and supporter models, while only the bootstrap trained weights will be saved. The resulting predicted probabilities are reduced using the arithmetic coder.

The main limitation of DeepZip and LSTM-compress is that they are exclusive to compress only particular data types and domain applications (i.e. sequential data such as textual and genomic data). As those RNN-based predictors follow a supervised dataset-driven learning-based mechanism, their models’ architecture is tuned particularly for specific data types (i.e. pre-trained for each dataset). Moreover, since the training involves only learning independently from existing data, thus DeepZip and LSTM-compress models are inapplicable for general-purpose compression. Also, DeepZip, LSTM-compress, and DZip have a substantial limitation that impractically influences encoding/decoding speed and computation time. Due to hardware or platform limitations (e.g. Keras [3]), these methods are forced to run their implementations in a deterministic environment to guarantee identical lossless outcomes. This issue restricted these codecs from utilising the GPU power, as it is only visible for their current implementations to run within a deterministic environment by operating on a single CPU thread.

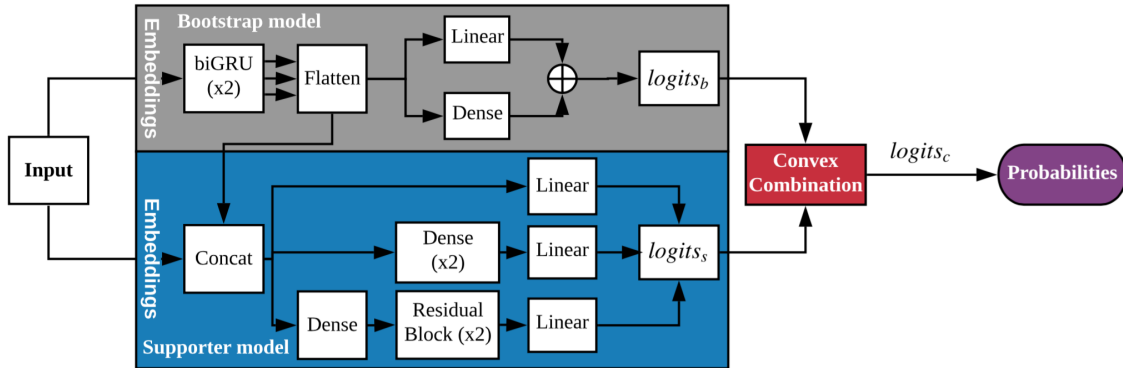


Figure 2.23: Combined model architecture consisting of bootstrap and supporter models. Dense layers correspond to fully connected layers with ReLU activation. Linear layers are also fully connected layers but do not incorporate a non linear transformation. Concat block denotes concatenation of all the input vectors [3].

2.4.5.3 Super-Resolution (SR) Compression Methods

Most of the existing contributions utilising the image Super-Resolution (SR) scheme address lossy compression quality. Within such a task, a model gradually reconstructs multi-level images from low- to super-resolution while enhancing overall visual appearance. Basically, given a low-resolution image, the SR model is expected to produce a high-resolution level through the learned representations. Various learning-based models demonstrate effectiveness by utilising such a mechanism while achieving state-of-the-art results using numerous diverse architectures such as CNN-based [151, 152], generative-based [153, 154], or likelihood-based methods [148]. Cao et al. designed an efficient lossless image compression coder with the **Super-Resolution based Compression (SReC)** strategy [4]. Their method encodes raw pixels that form the initial version with a low-resolution level. In contrast, higher levels are iteratively predicted while conditionally based on the probability of their low-level versions, as shown in Fig. 2.24. Predictions are produced in parallel as each four output pixels are grouped independently of the rest outputs at the same level. Each probability is then individually compressed to a lower bitrate by an entropy coder. Only the probability of three higher-resolution images is autoregressively predicted while leveraging SR models. The overall network architecture used in this work is a simple ResNet-like network design.

Another lossless codec that practically applied a full-resolution image compression scheme known as **L3C** was introduced by Mentzer et al. [32]. Similarly to the SReC codec, L3C is practically a fully parallelisable learning-based lossless codec. However, L3C is a probabilistic model that follows a hierarchical scheme whereby three forward passes are applied to predict all pixels. Unlike the well-known lossy autoregressive discrete probabilistic models (i.e. PixelCNN [155], PixelCNN++ [156], and Multiscale-PixelCNN [157]), the presented lossless codec jointly learns and produces full image distribution instead of likelihood over a pixel distribution. Moreover, their proposed probabilistic model contains feature extractors and predictors trained to model images and auxiliary features in a hierarchical format. An adaptive arithmetic coder is then involved in obtaining the compressed bitstream. Formally, given an image $x = z^{(0)}$, the total number of hierarchical scales utilised by L3C is denoted by S , wherein for each scale $s \in S$, a corresponding feature extractor $E^{(s)}$, and a predictor $D^{(s)}$ are employed to predict distribution p . The feature extractors $E^{(s)}$ produced quantised hierarchical feature representations indicated as $z^{(1)}, \dots, z^{(S)}$ for all levels. Given the joint distribution of $p(z^{(0)}, z^{(1)}, \dots, z^{(S)})$ with the auxiliary feature at a low level, a

non-autoregressive predictor $D^{(s)}$ will produce a summary feature $f^{(s)}$ for the predictor $D^{(s+1)}$ in the following larger-scale level.

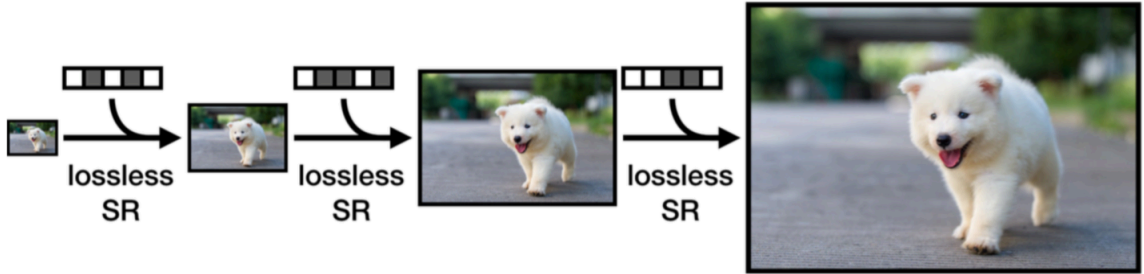


Figure 2.24: Model Overview. We propose lossless image compression through super resolution (SR). Our method first encodes a low-resolution image efficiently, and then leverages SR models to efficiently entropy-code the high-resolution images [4].

2.4.5.4 Transform-based Compression Methods

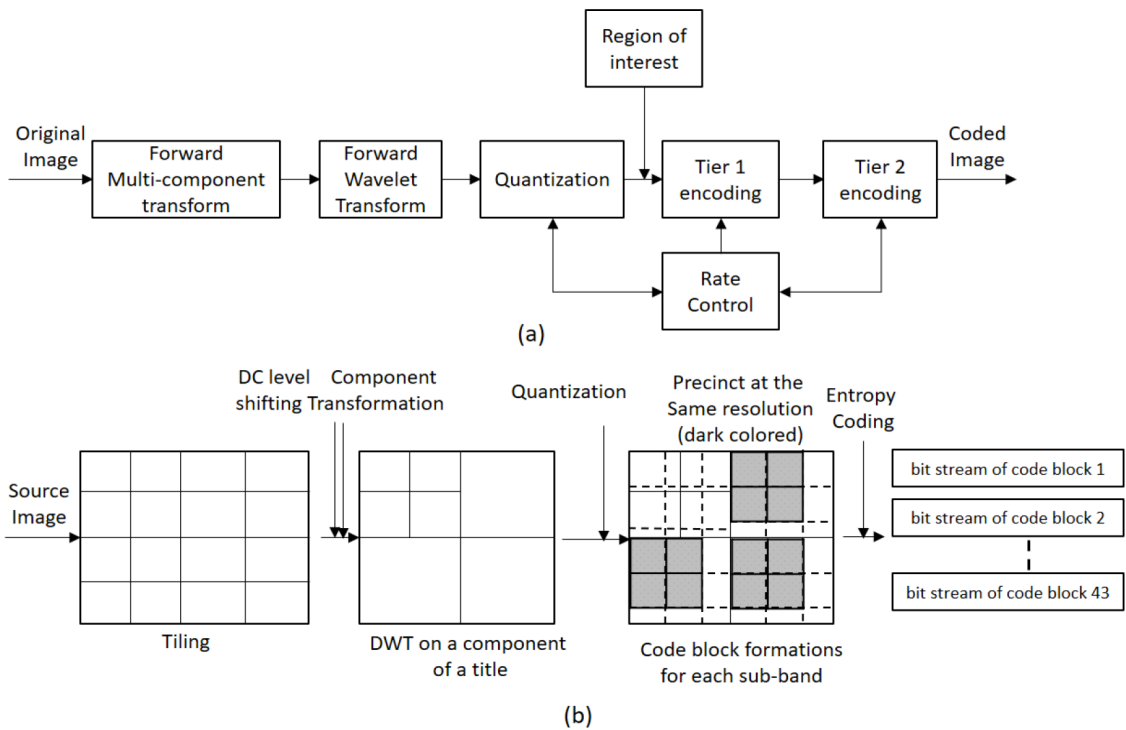


Figure 2.25: (a) Block diagram of the JPEG 2000 encoder algorithm. (b) Dataflow. [5]

Briefly, a transform-based algorithm converts an image/data spatial domain into a frequency domain, making it more compressible. The principle is to mathematically transform input spatial data to a more compact set of coefficients and then apply reduction steps such as quantisation or arithmetic coding. A well-known transform-based image coder is **Joint Photographic Experts Group 2000 (JPEG2000)** [142,158], which applies a DWT –(refer to subsection 2.3.2.5.2 for more details). This standard wavelet-based image compressor was designed to supersede the original DCT-based JPEG standard as the contributions such as memory efficiency, lossy to lossless coding, define a Region of Interest (ROI), and support for continuous-tone to bi-level images. The applications of this compressor extend to include not only 2D images but also volumetric imaging and wireless. Generally, JPEG2000 involves many stages within its compression framework, similar to most transform-based compressors, including tiling, colour component transform, discrete wavelet transform, quantisation, and entropy coding. Therefore, it empirically achieves a high compression ratio but yields more complicated computational complexity than other JPEG standards. JPEG2000 can offer both lossy or lossless compression qualities based on the wavelet transform and the quantisation performed. When examining the block diagram of JPEG2000 in figure 2.25 [6], an initial classification of reduction types for each block can be noticed whereby spatial redundancies, visual redundancies, coding redundancies are reduced, respectively. The image tiling principle was introduced to reduce the memory requirements while independently applying decoding to each tile. Typically, each image’s tile dimensionality is to the power of two except for the image edges. Moreover, multiple levels of quality or resolutions can flexibly be used by employing the tile concept. An essential element or region of interest would be placed in higher bit-planes to emphasise the Differential Coding (DC) component and assign a considerable coefficient value to be transmitted in better quality. A dyadic wavelet transform will be filtering the image at each subband while repeatedly subtracting the outcome from the original as illustrated in Fig. 2.26. Eventually, only the resulting difference formed as a single tile will need to be coded.

JP3D is an extension of JPEG2000 (known as Part 10 extension), which explicitly upgrades numerous of the key features from 2D into 3D space [7]. Such utilities include applying tiles, precinct code-blocks, resolution scalability, region-of-interest, and the 3D-DWT wavelet transform to the volumetric spatial dimension. Other z-dimension enhancements contain wavelet decomposition structures hierarchically. An illustration of Sub-band partitioning into dyadic-sized code-blocks is presented in figure 2.27 [7].

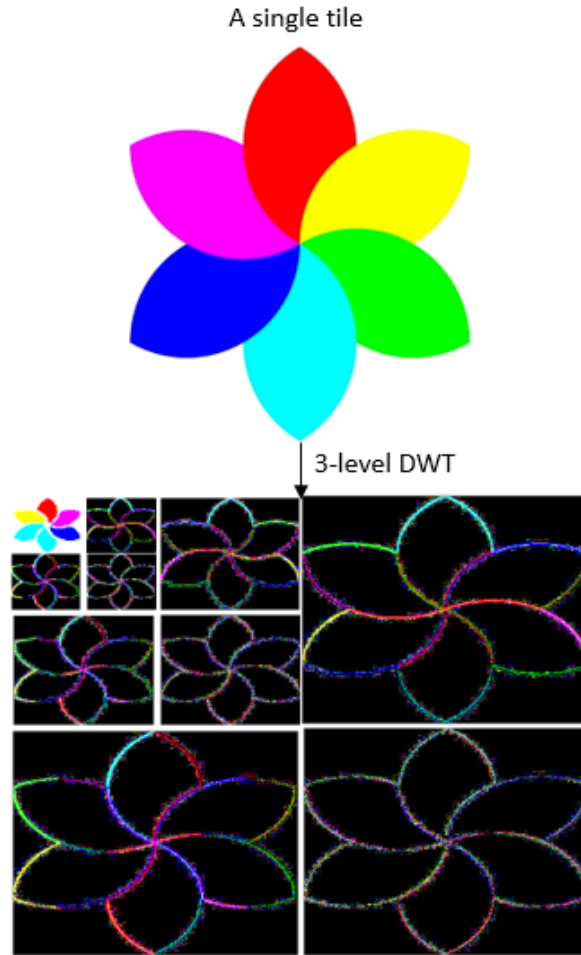


Figure 2.26: Dyadic decomposition of a single tile [6].

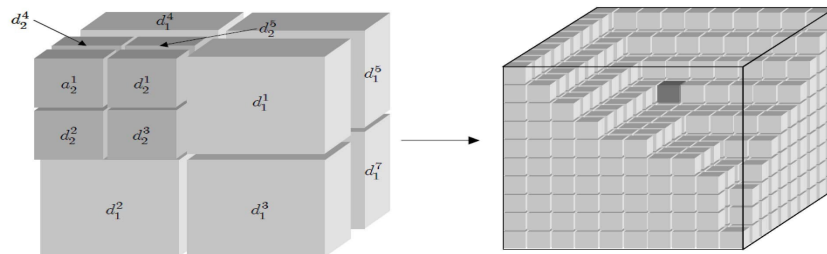


Figure 2.27: Code-block partitioning of a wavelet volume (tile). The dark cube in the right volume represents a code-block [7].

2.4.6 Prediction-based Compression Methods

By examining the existing literature on prediction-based codecs, it is noticeable that there is a lack of extensive studies that explore and review both classical and learning-based lossless approaches in terms of the utilised spatial context and predictive patterns. Thus, this subsection aims to intensively cover all related aspects of lossless prediction-based codec, including classification of utilised spatial context, overall framework encoding/decoding procedures, and highlighting the current research trending techniques. The following two subsections will emphasise the state-of-the-art classical (non-learned) and learning-based compression methods that apply lossless prediction-based paradigm within the current literature.

A **predictive-based coding** is a particular compression algorithm that utilises spatial information from the data context. Such a methodology allows inputting prior observations to the encoder/decoder in which a prediction paradigm is performed to output the next value. As image's pixels/voxels tend to be highly correlated, this coder gains reduction by removing spatial redundancy between them. When the codec conducts a mathematically reversible set of compression steps and recovers original data without any loss, the compression quality is lossless, while irreversible operations (e.g. quantisation) yield lossy quality. The compression processes in a predictive-based codec typically consist of the prediction phase, residual errors computation phase, and coding redundancy removal phase. Some predictive-based models may involve contextual modelling or error modelling for further bit-reduction. A predictor decorrelates the spatial information from the previous pixels to yield the next pixels at the prediction phase. Then, the prediction/residual errors are then calculated as the difference between the actual and predicted values, which usually has a distribution similar to Gaussian normal distribution. Lastly, the coding redundancies are removed losslessly by an Entropy coder producing the compact form of the data. Generally, the more precise a predictor is in exploiting spatial features representations, the closer its predictions to actual values become, resulting in negligible residual errors (i.e. close to zero) with a more compressible coding signal. A general block diagram for a lossless predictive-based compression framework with a detailed overview of the encoder and the decoder schemes is proposed in figure 2.28.

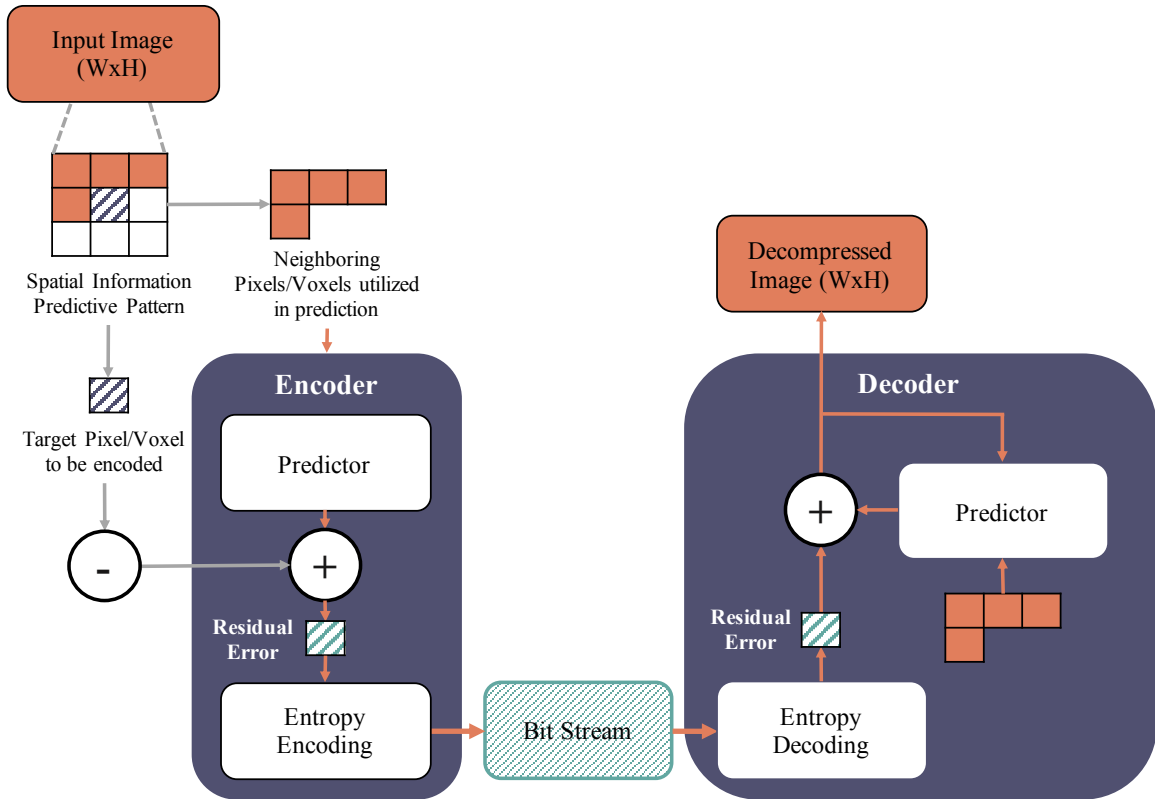


Figure 2.28: A general block diagram for a lossless predictive-based compression framework with a detailed overview of the encoder and the decoder blocks.

Formally, given an image \mathcal{I} , let $\mathcal{S} \subset \mathcal{I}$ where \mathcal{S} denotes a set of pixels within the image, consisting of finite paired of (previous pixel(s), output pixel(s)) defined as $\mathcal{S} = \{x_i, y_i\}_{i=0}^n$, where n is the total number of pixels in \mathcal{I} . For each x_i input value(s), a lossless predictor-based codec $\mathcal{P}(x_i)$ is generally expected to predict output value(s) \hat{y}_i , by exploiting the spatial redundancies using linear or nonlinear mechanism/function (e.g. hand-crafted computations, or learnable mapping functions). The residual error values r_n are computed as $r_i = y_i - \hat{y}_i$ for each prediction value(s). An entropy coder is involved to reduce the coding redundancies to lower bit-rate. When decoding at a receiver side, the entropy coder decodes the residual error r_n values. Then, for each prediction value \hat{y}_i , the matching prediction error value r_i is summed using $y_i = r_i + \hat{y}_i$ to recover the original value y_i .

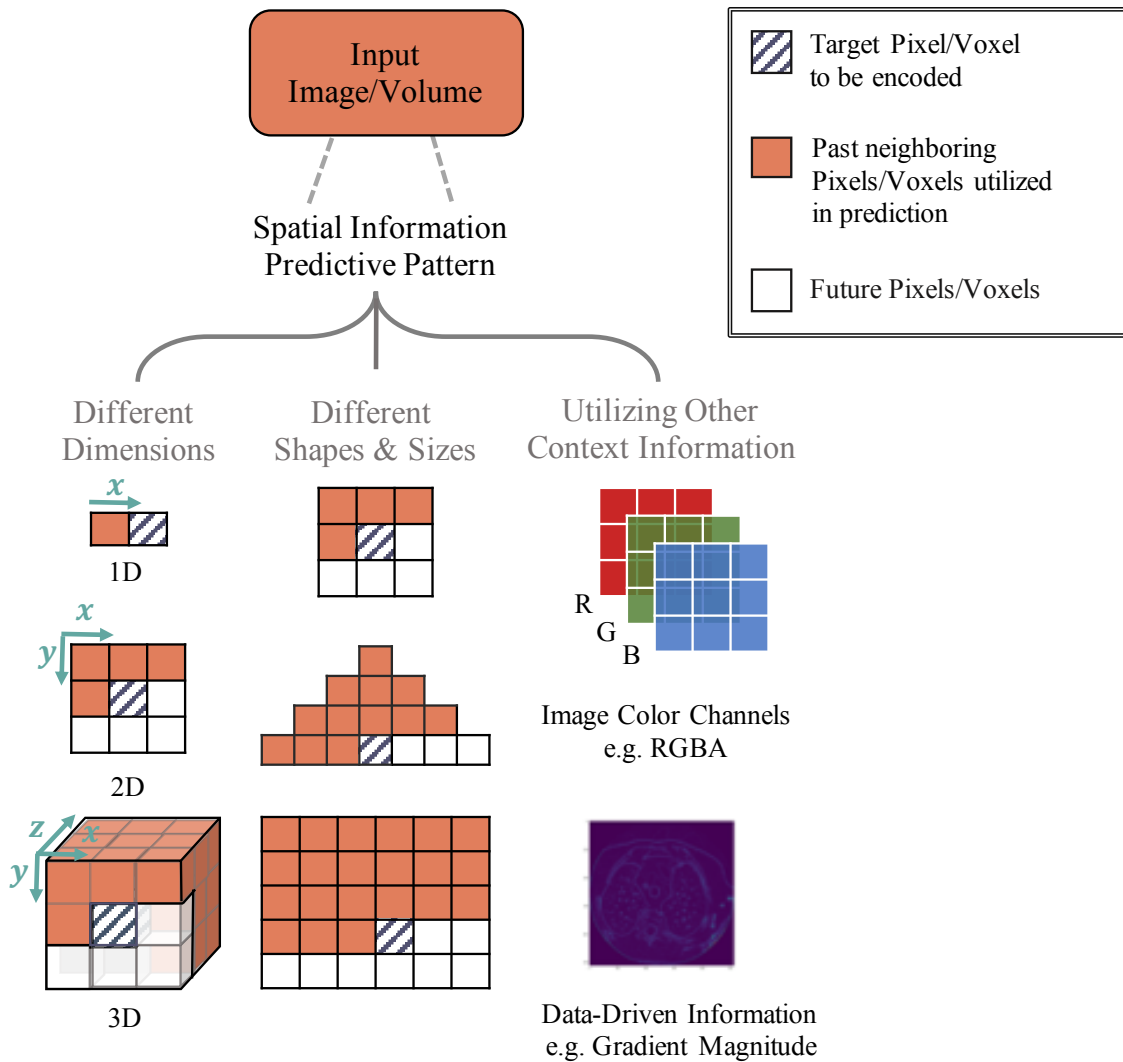


Figure 2.29: A general illustration of the spatial information and predictive patterns that a prediction-based compression extracts and manipulates when compressing image/volume data. A classification of the numerous options utilised in the compression literature is proposed, including different dimensions, shapes, block sizes, and other data-driven features or colour channels of the available spatial information. Patterns' pixels/voxels are colour mapped based on their functionalities when employed by the predictor. For instance, past neighbouring pixels/voxels (coloured in Orange) are input and used in the prediction. However, the pixel/voxel coloured in striped Purple represents the target location currently encoded or decoded. Lastly, White pixels/voxels are future pixels/voxels.

Based on the existing literature of the prediction-based codec, a broad illustration of the numerous spatial information and predictive patterns extracted and manipulated when compressing image/volume data is demonstrated in figure 2.29. These numerous

options are classified based on the input sequence’s dimensions, shapes, block sizes, and utilisation of other available spatial information such as data-driven features or colour channels. As such contextual information influences and assists a predictor in exploiting spatial redundancies, many of the existing predictive-based methods employ different variations with different block sizes and sequence lengths, spanning both classical and learning-based approaches. These causal neighbourhood sequences are linearly or nonlinearly combined and decorrelated by various prediction-based approaches while reducing bitrates considerably. Some predictor codecs may also utilise spectral information for further exploiting the local structures. For all the examples in the same figure, patterns’ pixels/voxels are colour mapped based on their functionalities when employed by the predictor. For instance, past neighbouring pixels/voxels (coloured in Orange) are fed and used in the prediction. However, the pixel/voxel coloured in striped Purple represents the target location currently encoded or decoded. Lastly, White pixels/voxels are future pixels/voxels.

2.4.6.1 Classical Prediction-Based Methods

This subsection will present some of the current literature’s prediction-based classical codecs that perform lossless compression quality. Example of such codecs are Joint Photographic Experts Group-Lossless (JPEG-LS) [13], Context Based Adaptive Lossless Image Codec (CALIC) [134], (3D-CALIC) [136], (M-CALIC) [137], High Efficiency Video Coding (HEVC) [9], Minimum Rate Predictors (MRP) [135], and 3D-MRP [10].

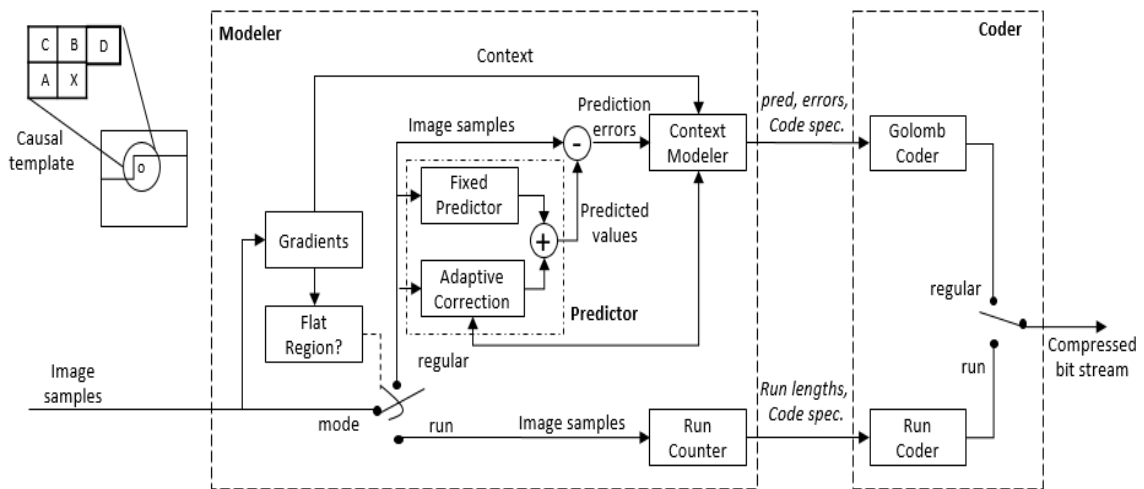


Figure 2.30: Block diagram for the JPEG-LS encoder [8].

Selection-value	Prediction
0	No prediction
1	A
2	B
3	C
4	$A + B - C$
5	$A + (B - C)/2$
6	$B + (A - C)/2$
7	$(A + B)/2$

Table 2.4: The mode-selection options utilised in the JPEG-LS predictor [13]

The **Joint Photographic Experts Group-Lossless (JPEG-LS)** is a well-known predictive compression standard that involves a mode-selection method similar to the LOw COmplexity LOSSless COmpression for Images (LOCO-I) algorithm [13]. Unlike JPEG2000, the JPEG-LS follows a different encoding mechanism that allows lossless and near-lossless compression qualities to reduce continuous-tone, grayscale and colour still images. The main intention of designing this algorithm is to provide compression with low complexity while ensuring lossless reconstruction. JPEG-LS applies a predictive paradigm that utilises some of the input structures, precisely the three nearest neighbouring pixels (i.e. A, B, and C), to determine the reduction mode conditionally based on the context within (see figure 2.30 for more details). Such a mode-selection scheme allows flexibly encoding each pixel to adaptively vary given the conditional probabilities from its context while capturing high-order dependencies. In order to obtain reasonable decorrelation, this algorithm presents eight options for mode-selections, as shown in table 2.4. For instance, three options maintain a 1D predictor linear operations, while four other options facilitate 2D operations, such as a primitive test of vertical or horizontal edges detections. The JPEG-LS prediction mechanism is built on the Median Edge Detection (MED) or LOCO-I predictor, detecting simple primitive edges (i.e. horizontal or vertical edges). A direct switch between options is used by examining the nearest neighbouring pixels, whereby B represents the vertical edge, A represents a horizontal edge, and the median is selected if no edge is detected. A formal overview of the JPEG-LS predictor [13] is defined as:

$$P_{JPEG-LS}(x, y) = \begin{cases} \min(A, B), & \text{if } C \geq \max(A, B), \\ \max(A, B), & \text{if } C \leq \min(A, B), \\ A + B - C, & \text{otherwise} \end{cases} \quad (2.37)$$

Then, context modelling encodes the current prediction residual by exploiting texture patterns and local activity through the utilisation of local variations in pixel values (i.e. local gradients) calculated as:

$$\begin{aligned} g_1 &= D - B \\ g_2 &= B - C \\ g_3 &= C - A \end{aligned} \tag{2.38}$$

Such local gradients allow JPEG-LS to control the statistical behaviour of prediction errors reflected by the activity level (i.e. smoothness, and edginess) surrounding the target sample in that particular context. Lastly, the prediction errors will be losslessly compressed using the Golomb codec.

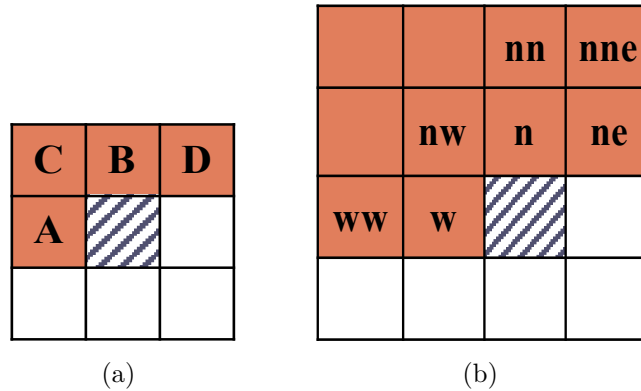


Figure 2.31: An illustration of utilised local neighbouring causal pixels in JPEG-LS predictor (a) and CALIC predictor (b).

Another compressor that involves a prediction-based paradigm with two phases of prediction and residual is the **Context-based Adaptive Lossless Image Codec (CALIC)** [134]. Similar to JPEG-LS, within the given context, CALIC utilises a gradient-based prediction scheme, where adjusted prediction coefficients are estimated from the local gradients. Based on the neighbourhood contextual information of the current pixel and seeking a better reduction, the encoding may adaptively change if the local structure contains horizontal or vertical edges. Particularly, CALIC computes an edge descriptor to determine seven edge types, including edge’s direction and strength as calculated in equations 2.40. Unlike JPEG-LS, CALIC applied a complex context conditioning based on a Gradient-Adjusted Predictor (GAP) nonlinear predictor, utilising a causal neighbourhood of six pixels. With the GAP predictor, a set of predefined heuristic thresholds are used to determine whether the local context is weak, regular, sharp, horizontal edge, vertical edge, or smooth area.

Formally, let $I(x, y)$ denote the current pixel position desired to be encoded. Let denote each of its local neighborhood pixels with: $n = I(x, y - 1)$, $w = I(x - 1, y)$, $nw = I(x - 1, y - 1)$, $ne = I(x + 1, y - 1)$, $ww = I(x - 2, y)$, $nn = I(x, y - 2)$, and $nne = I(x + 1, y - 2)$, as labeled in Fig. 2.31(b). An edge descriptor d is estimated by the local gradients of the current pixel to determine the edge's strength, where g_h denote the horizontal edge and g_v vertical edges, defined as:

$$d = g_v - g_h, \quad \text{where} \begin{cases} g_h = |w - ww| + |n - nw| + |n - ne| \\ g_v = |w - nw| + |n - nn| + |ne - nne| \end{cases} \quad (2.39)$$

The CALIC predictor [134] is defined as:

$$P_{CALIC}(x, y) = \begin{cases} w, & \text{if } (g_v - g_h) > 80 & \{\text{sharp horizontal edge}\} \\ n, & \text{elseif } (g_v - g_h) < -80 & \{\text{sharp vertical edge}\} \\ \text{else} \\ \hat{M}_p = \frac{(w+n)}{2} + \frac{(ne-nw)}{4} \\ \frac{(\hat{M}_p+w)}{2}, & \text{if } (g_v - g_h) > 32, & \{\text{horizontal edge}\} \\ \frac{(3\hat{M}_p+w)}{4}, & \text{elseif } (g_v - g_h) > 8, & \{\text{weak horizontal edge}\} \\ \frac{(\hat{M}_p+n)}{2}, & \text{elseif } (g_v - g_h) < -32, & \{\text{vertical edge}\} \\ \frac{(3\hat{M}_p+n)}{4}, & \text{elseif } (g_v - g_h) < -8, & \{\text{weak vertical edge}\} \end{cases} \quad (2.40)$$

The residual error is then encoded using a context-adaptive variable-length entropy coding method.

3D-CALIC expands the functionality of CALIC to higher-dimensional data wherein statistical redundancies of inter-band and intra-band are reduced [136]. In this coder, the non-linear spectral predictor utilises not only the local structure in the current band but also values from its reference band. Exploiting correlation within neighbouring pixels in the same band is known as spatial (intra-band) correlations. In contrast, a correlation between pixels in adjacent bands is known as spectral (inter-band) correlations. Such methodology allows exploiting both inter-band and intra-band based on the local neighbourhoods correlations. Mainly, 3D-CALIC adaptively changes between the inter-band and intra-band predictors based on the value of local correlation coefficients between current band X and reference band Y . Then, a context-based entropy reduces the statistical redundancies by utilising residual correlations and spatial or spectral context. An improved version that is efficiently applied as a spectral

decorrelator is **M-CALIC** [137], where the prefix “M” stands for the multiband nature of the spectral prediction. This codec offers both lossless and near-lossless compression qualities while reducing Band-Interleaved-by-Line (BIL) data formats such as satellite or aircraft images. M-CALIC exploits spectral correlations not only in multispectral data but also hyperspectral data with (16 bit-depths) utilising modified spatial predictors and complex spectral predictors with an optimised CALIC-based compression engine. The novel predictor in this coder involves two previous bands in the current reference line while optimising model parameters and quantisation thresholds, gaining better reduction than 3D-CALIC and previous versions.

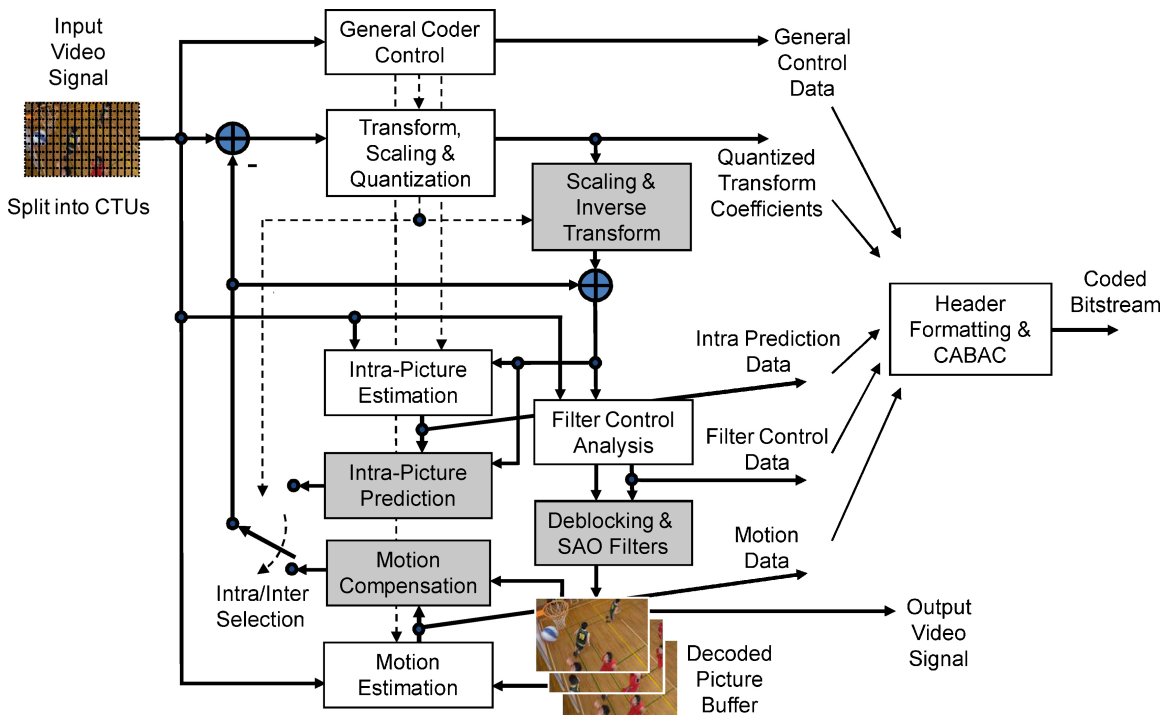


Figure 2.32: Typical HEVC video encoder (with decoder modeling elements shaded in light gray) [9]

High Efficiency Video Coding (HEVC) is a well-known standard video coding that supports high bit-rate reduction while retaining perceptual video quality [9, 159]. This codec proposes many technical features and characteristics for both lossy and lossless compression qualities. The domain applications of such compression framework are not restricted to video reduction but extend to include other domains such as volumetric medical scans, broadcast of high definition (HD) TV signals over satellite, cable, video content acquisition and editing systems, security applications, and Internet and mobile network video. The compression process starts by subdividing images into

non-overlapped block-shaped regions at which a prediction paradigm will be applied, whereas each block can be encoded differently based on the motion information. Like JPEG2000, HEVC supports the tiling concept for more efficient computations and parallel processing. By partitioning images into smaller rectangular regions (tiles), independently decodable parallel computations can be performed. In HEVC, finding the best-matched block is determined by the motion information of the frame. As this codec handles higher dimensional data distributions, the prediction scheme applied not only utilised pixels in the current frame but also excessive frames. When the motion signal is determined from the current frame, it is known as (Intra), while the previously encoded frames are known as (Inter). Similarly to the other predictive-based compressors, after prediction, the residual error is computed and transformed into transform blocks, wherein the coding redundancy is reduced using an entropy coder. A general block diagram of the HEVC video coder is illustrated in figure 2.32.

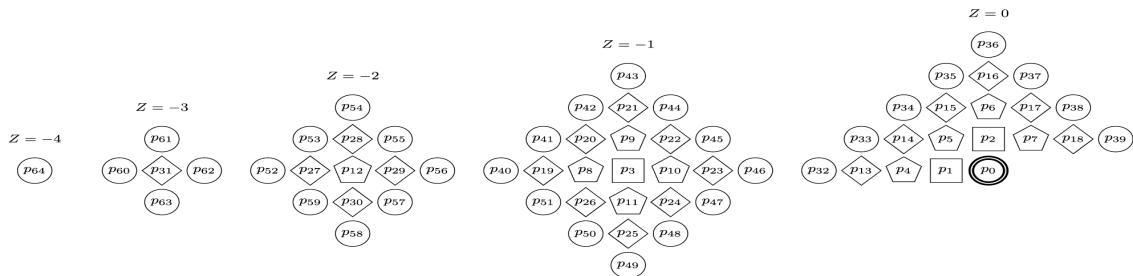


Figure 2.33: 3D-shaped support used for linear prediction in the proposed 3D-MRP algorithm. Pixels located at previous frames are indexed by the Z coordinate, with $Z = 0$ corresponding to the current frame. [10]

Minimum Rate Predictors (MRP) supports an adaptive block-based prediction scheme whereby each block can be encoded by a different predictor classified by a set of coefficients [135]. Generally, the MRP algorithm will be iteratively optimising three core functions while minimising a specified cost function. These functionalities include the determination of minimum rate predictors, estimation of quantisation thresholds,

and classification of the blocks. The predictor linearly utilises some small causal neighbourhood pixels (i.e. 5 pixels) previously encoded with Manhattan distance to encode the current reference pixel. The prediction error will be minimised using context modelling that infers the Probability Density Function (PDF) of the error followed by an arithmetic coding. An enhanced version, which extends the MRP functionality and characteristics to 3D space known as **3D-MRP**, was proposed by Lucas et al. [10]. 3D predictors, 3D-block octree partitioning and classification, and volume-based optimisation for high bit-depth volumetric medical dataset reduction are supported within this implementation. The algorithm subdivides volume data into fixed-size 3D blocks, whereby the 3D-predictor will independently be applied while specifying a class and threshold per block. The enhanced 3D-predictor exploits not only spatial redundancies within the current frame but also redundancies from prior encoded frames (i.e. inter-frame redundancies) within the 3D block. The maximum number of casual voxels utilised by the predictor is 64 voxels, with a maximum Manhattan distance of 4 pixels horizontal, vertical, and axial directions, as shown in Fig. 2.33. The classification strategy supports hybrid 2D and 3D types whereby octree block partitioning is applied based on the given region context. The intention is to evaluate whether a single class can efficiently represent the 3D block or multiple classes corresponding to each of the 2D blocks alternatively utilised. Similar to the original MRP algorithm, quantisation of thresholds using enhanced context modelling and arithmetic coding to reduce residue coding is applied.

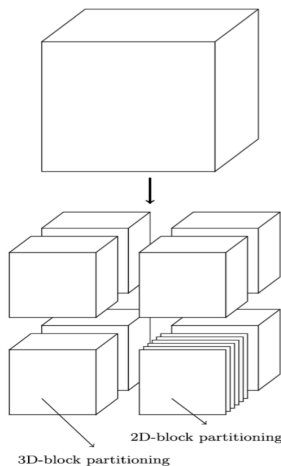


Figure 2.34: Octree-based partitioning of a 3D-block, with one of the subblocks (in the bottom-right corner) being further divided into independent 2D-blocks, as proposed by the hybrid classification step of 3D-MRP algorithm. [10]

2.4.6.2 Learning-Based Prediction Methods

The shifting toward deep learning strategies instead of traditional codecs is due to their increased capabilities and robustness in understanding and capturing nonlinear domains and data distributions. Compared to the classical methods, learning-based approaches can be adjusted to many dataset types while remarkably beating traditional outcomes. Many well-known classical codecs utilise hand-crafted features or linear combinations schemes when reducing data size, restricting their overall performance. In contrast, deep learning methods demonstrate higher flexibility in representing nonlinear distributions and likelihood estimations while learning arbitrarily complex mappings and patterns. Overall, innovations and contributions to learning-based approaches from great potentials and considerably promising research direction. The current deep learning methods are outperforming state-of-the-art classical methods in solving numerous domain problems, including data compression for both lossy and lossless qualities [160–162]. Within the data compression field, the research contributions utilising deep learning methods offer either novel strategies or hybrid coding solutions replacing some components of traditional codecs with DL alternatives. Such components can be the predictor, quantiser, and entropy coder. This subsection will highlight the current research contributions within lossless compression using deep-learning prediction-based techniques.

A novel DL-based prediction method combined with error modelling and a context-tree based bit-plane (CBP) codec for residual error encoding was proposed in [138]. The applications for this codec includes reducing photographic images, lenslet images, and video frames losslessly. A pixel-wise prediction scheme is used, where large local causal neighbourhood pixels are utilised for predicting the current pixel (i.e. size of $(b+1) \times (2b+1)$ causal neighbourhood, which selects up to a distance of b pixels from the current position). A dual prediction strategy is employed wherein the prediction of any pixel is made with some state-of-the-art prediction methods and enhanced by predicting its residual with a CNN-based model. Also, an enhanced version of CALIC’s context modeller is employed as an error modelling unit. Furthermore, the modelled error is then further reduced by an entropy coder.

A novel macro-pixel prediction method based on Convolutional Neural Network (CNN) model (MP-CNN) was introduced by Schiopu, and Munteanu [163], specifically for reducing large light field images losslessly. Their predictor utilises a volume of six immediate causal macro-pixels neighbourhood to predict the current macro-pixel value. The residual errors are encoded with a new context modelling method built on

the CALIC codec to utilise the macro-pixel structure in lenslet images. The model architecture for this lenslet image codec consists of 57 convolutional layers.

A block-wise prediction using CNN-based with multi-resolution design is proposed by Schiopu et al. [140] and known as Angular intra-Prediction Convolutional Neural Network (AP-CNN), especially for lossless video coding applications. The novelty in their methodology is replacing HEVC standard intra-prediction schemes with learning-based CNN prediction methods. Better bit reductions (i.e. rate saving of 5% over HEVC) are accomplished by efficiently enhancing some HEVC components with alternative learning CNN-based blocks. A novel residual block estimator along with 33 angular learning-based inter-prediction replacement modes are presented. The prediction mechanism operates in pixel-wise and block-wise (i.e. 4×4 and 8×8 block sizes) paradigms while seeking better reductions for different image types. The primary drawback of this hybrid video coder is the higher computation cost of inferencing compared to the original HEVC implementation, which is expected given the high run-time required by the CNN-based model.

A learning-based model that uses not only spatial information but also spectral information for lossless hyperspectral images coding was introduced by Jiang et al. [164]. Their proposed shallow neural network (i.e. only two-hidden layers) operates as an adaptive filter, where feature extraction and predictive filtering are performed. The architecture comprises two parallel NN concatenated, wherein each is fed with different input formats (i.e. spatial and spectral contexts). Thus, this codec is known as Concatenated Shallow Neural Network (CSNN). The model's output predicts the target pixel; then, the prediction error is computed and encoded using an entropy coder (i.e. Golomb Rice Codec). Their presented methodology implies low computational cost as it does not demand pre-training and has a relatively small model. Compared to other state-of-the-art codecs, CSNN achieves higher compression results.

A comprehensive exploration of the current literature for lossless predictive-based codecs is presented in Table 2.5, including classical and learning-based approaches with a detailed overview of prediction patterns (i.e. the number of causal neighbourhoods utilised), spatial dimensions, domains applications, and methodology details.

Predictive Coding	Method Type	Number of Causal Neighborhood Utilised	Dimensions of the Causal Neighborhood Utilised	Compression Quality	Data Compression Domains	Methodology Details	
JPEG-LS [13]	Classical	3	2D	Lossless	Image (16 bit-depths)	LOCO-I mode-selection predictor based on linear combinations of small causal neighbourhood	
HEVC [9]	Classical	Coding Tree Unit (CTU), which can vary in size from 8×8 to 64×64 pixels	3D	Lossless/Lossy	Video and Medical Volumes (16 bit-depths)	HEVC partitions frames into smaller rectangular regions (tiles), whereby Inter/Intra predictor reduces temporally or spatially neighbouring redundancies. Finding the best-matched block is determined by the motion information within the frame. After prediction, the residual error is computed and transformed into transform blocks, wherein the coding redundancy is reduced using an entropy coder.	
MRP [135]	Classical	The K previous encoded reference pixels, ordered by an increasing Manhattan distance from the unknown pixel. In MRP, $K=5$	2D	lossless	Still Images (8 bit-depths)	MRP supports an adaptive block-based prediction scheme whereby each block can be encoded by a different predictor classified by a set of coefficients. The prediction error will be minimised using context modelling that infers the Probability Density Function (PDF) of the error followed by an arithmetic coding.	
3D-MRP [10]	Classical	Total of 64 voxels, with maximum Manhattan distance of 4 pixels in the horizontal, vertical, and axial directions	3D	lossless	Volumetric Medical Images (16 bit-depths)	3D-MRP extends and enhances the original functionalities of MRP to 3D space while supporting the reduction of volumetric medical images for both 8 and 16 bit-depths. The 3D-predictor in this version utilised 3D causal neighbouring with 64 voxels.	
CALIC [134]	Classical	6	2D	lossless	image (8 bit-depths)	CALIC computes an edge descriptor to determine seven edge types, including its direction and strength. A complex context conditioning based on a GAP nonlinear predictor utilises a small causal neighbourhood. The prediction error is then encoded using a context-adaptive variable-length entropy coding method.	
3D-CALIC [136]	Classical	Neighbourhoods pixels from two bands to predict the current sample pixel	3D	lossless	multi-spectral images such as color images, remotely sensed images, and satellite images (16 bit-depths)	3D-CALIC contains a non-linear spectral predictor that uses values in a reference band to predict the current band. Such methodology allows exploiting both inter-band and intra-band based on the local neighbourhoods correlations. A context-based entropy reduces the statistical redundancies by utilising residual correlations and some spatial or spectral context.	
M-CALIC [137]	Classical	Neighbourhood pixels from the two previous image's rows	3D	Lossless/Near-Lossless	Satellite images or Aircraft images (16 bit-depths)	An improved version of CALIC-based coder, which proposes a novel multiband spectral predictor at which spectral data are effectively decorrelated. Within this predictive coder, a linear combination of the two previous bands is involved in the current reference line while optimising model parameters and quantisation thresholds selections.	
Concatenated Shallow Neural Network (CSNN) [164]	DL-based	Spatial context (12 neighbouring pixels from current and the 2 previous bands) spectral context (4 pixels from previous bands co-located with the current pixel)	3D	Lossless	Hyperspectral images	A shallow neural network (i.e. only two-hidden layers) operates as an adaptive filter, where feature extraction and predictive filtering are performed. The architecture comprises two parallel NN concatenated, wherein each is fed with different inputs (i.e. spatial and spectral contexts). The model's output predicts the target pixel; then, the prediction error is computed. Lastly, the residuals are encoded using an entropy coder (i.e. Golomb Rice Codec).	
DeepZip [145]	DL-based	A sequence is divided into overlapping segments of length $K + 1$ (shifted by one), where K represents the previously encountered symbols. In DeepZip, K was chosen to be 64.	Sequential Data	Lossless	Text and Genomic Data	DeepZip provides a lossless encoder-decoder framework wherein each model consists of two primary blocks: an RNN-based probability estimator and arithmetic coding.	
LSTM-Compress [146]	DL-based	Not specified	Sequential Data	Lossless	Text and Genomic Data	Like DeepZip, LSTM-Compress also utilises LSTM predictors as probability estimators to adaptively learn the source distribution while condensing the learned representations into more compact forms with an arithmetic coding unit.	
Dzip [3]	DL-based	The number of previous symbols used for prediction is set by default to $K = 64$	Sequential Data	Lossless	Variety of Real Datasets (regardless of the alphabet size), for example, Text, Genomics, Executable Files, Audio Data, and Double Precision Floating-point Datasets	A general-purpose lossless compressor uses NN-based statistical data modelling combined with arithmetic coding. During training, DZip utilises a hybrid training scheme at which a combination of semi-adaptive and adaptive training approaches are applied to two models, a bootstrap model and a supporter model.	
CNN-based Prediction (PredNN) [165]	DL-based	The causal neighbourhood $N(b)$ of size $(b + 1) \times (2b + 1)$ selects pixels up to the distance b from the current position. The last $b+1$ values on the last row are set to 0.	2D	Lossless	High-resolution Photographic Images	A novel pixel-wise conventional predictor that utilises different predicting mechanisms based on the textural region types. For highly textural regions with sharp edges, the traditional LOCO-I predictor is applied, while for flat areas with weak edges, the traditional CALIC predictor is used. Such a methodology allows this CNN-based model to recognise the local patterns and identify a feature vector. CALIC reference coder is then involved in reducing the coding redundancy.	
Residual-error prediction REP-CNN [139]	DL-based	The causal neighbourhood $N(b)$ of size $(b + 1) \times (2b + 1)$ selects pixels up to the distance b from the current position. The last $b+1$ values on the last row are set to 0.	2D	Lossless	High-resolution Photographic Images	Enhance the residual coding unit by utilising a reference codec based on CALIC and a novel residual error prediction model (REP-CNN).	
Macro-Pixel prediction (MP-CNN) [163]	DL-based	A volume of six macro-pixels of immediate causal neighborhood.	2D	Lossless	Large Light Field Images	A CNN-based predictor uses block-wise macro-pixels specifically for compression light field images losslessly. The prediction errors are then encoded by a reference CALIC codec adapted to perform macro-pixels of the lenslet images with a new context modelling method.	
CNN-Based Intra-Prediction [140]	DL-based	A block-wise (4×4 and 8×8 block sizes)	3D	Lossless	Video	A block-wise prediction using CNN-based with multi-resolution design is proposed. The novelty in their methodology is replacing HEVC standard intra-prediction schemes with learning-based CNN prediction methods.	
Novel Hybrid Coding [138]	Hybrid	DL-based	The causal neighborhood $N_b(x, y)$, of size $(b+1) \times (2b+1)$ which selects pixels up to the distance of b pixels from the current position. The last $b+1$ values in the last row are unknown at the decoder side. Thus, they are set to 0.	2D/3D	Lossless	High-resolution Photographic Images, Lenslet Images, and Video Frames	A novel DL-based prediction method with context-tree based bit-plane (CBP) codec for residual error coding. A pixel-wise prediction scheme is used, where large local causal neighbourhood pixels are utilised for predicting the current pixel. A dual prediction strategy is employed wherein the prediction of any pixel is made with some state-of-the-art prediction methods and enhanced by predicting its residual with a CNN-based model. Also, an enhanced version of CALIC's context modeller is employed as an error modelling unit.

Table 2.5: A comprehensive exploration of the current literature for lossless predictive-based codecs is presented, including classical and learning-based approaches with a detailed overview of prediction patterns (i.e. the number of causal neighbourhoods utilised), spatial dimensions, domains applications, and methodology details.

2.4.7 Challenges and Observations

- As only limited contributions have been recently made to the lossless learning-based compression, no comprehensive survey studies yet cover this particular literature. Most recent review papers focus on lossy learning-based techniques or other lossless classical-based compression methods.
- One of the main challenges for learning-based domain methods is the generalisability across unseen volumes, different medical modalities, and scanner settings (e.g. variations in HU's ranges based on beam's setting [55]).
- It would be interesting to notice research contributions that propose end-to-end compression frameworks seeking lossless compression quality potentially. For instance, iWave3D [166] that was recently introduced by Xue et al. iWave3D is an end-to-end Brain image compression based on learning-based 3-D Wavelet Transform.
- Naturally, it is challenging to develop an efficient combination of a likelihood-based model and entropy coding that is practical and computationally efficient while gaining a reasonable compression rate for lossy or lossless quality. However, some existing learning-based models demonstrate promising yet practically fully parallelisable implementations such as L3C [32] and SReC [4].

Overview of Materials, Losses and Metrics



Contents

3.1	Dataset Details	114
3.1.1	Dataset 1 - Private CT	115
3.1.2	Dataset 2 - Public CT	118
3.1.3	Dataset 3 - Public MRI	119
3.2	Loss Function	121
3.3	Evaluation Metrics	121
3.3.1	Compression Ratio	121
3.3.2	Compression Time	121

This chapter covers the experimental details shared within the following technical chapters of this thesis. Such information includes descriptions of dataset details, metrics used for evaluation, customized loss functions, and other experimental setups.

3.1 Dataset Details

All datasets utilised in this thesis consist of a set of DICOM files stored in 16 bit-depths grayscale images. The modality of the medical images belonging to any dataset is either Computed Tomography (CT) or Magnetic Resonance Imaging (MRI). More details on each of the used datasets are given in the following subsections.

3.1.1 Dataset 1 - Private CT

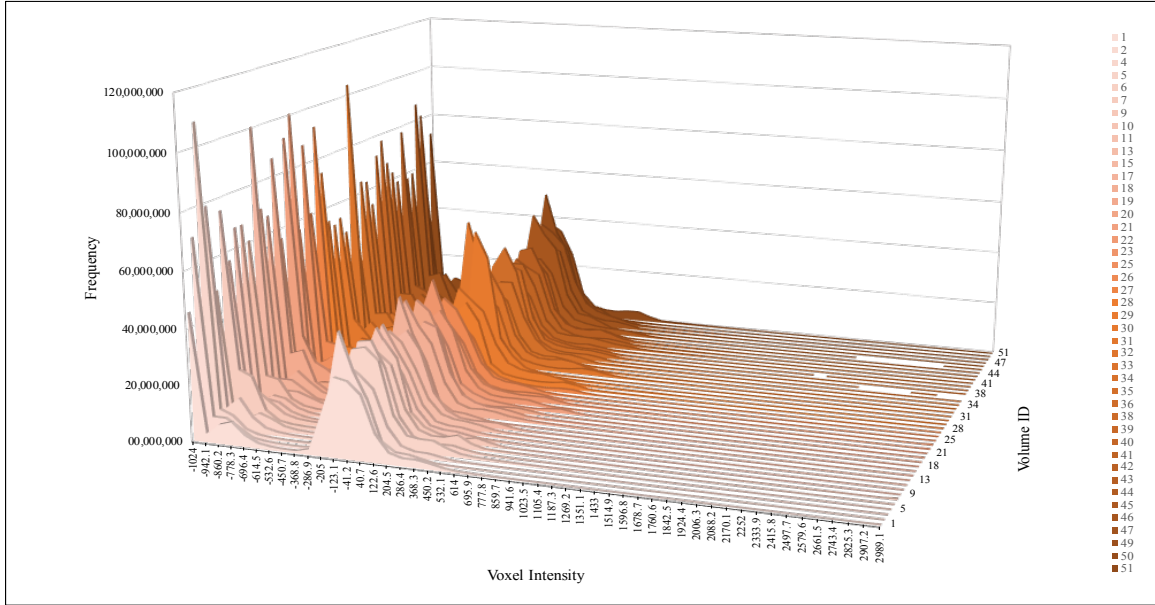


Figure 3.1: Overview of the voxel intensity histogram across all volumes (16 bit-depths) belonging to dataset 1.

This private dataset contains 43 high-resolution volumes generated by the same hospital and represents CT scans for a patient’s entire trunk, stored in DICOM format. All the scans have the same resolution (i.e. width and height) 512×512 ; however, they differ in the depth of the volume (i.e. number of frames per volume) $z \in [750, 1120]$. The slice thickness is 0.625 mm in all patients belonging to this set. The pixel spacing **varies** between patients $PS \in \{.488, .578, .625, .703\}$. Intensity values range from -1024 to 3071 (12 bits stored as 16 bits integer). An illustration of the voxel intensity histogram across all volumes (16 bit-depths) belonging to dataset 1 is shown in Fig. 3.1. Moreover, a detailed overview of dataset 1 characteristics is presented in table 3.1. Furthermore, some sample volumes (16 bit-depths) belonging to this set are presented in Fig. 3.2. This dataset was utilised in all the technical chapters of this thesis, including chapters 4, 5, 6, and 7.

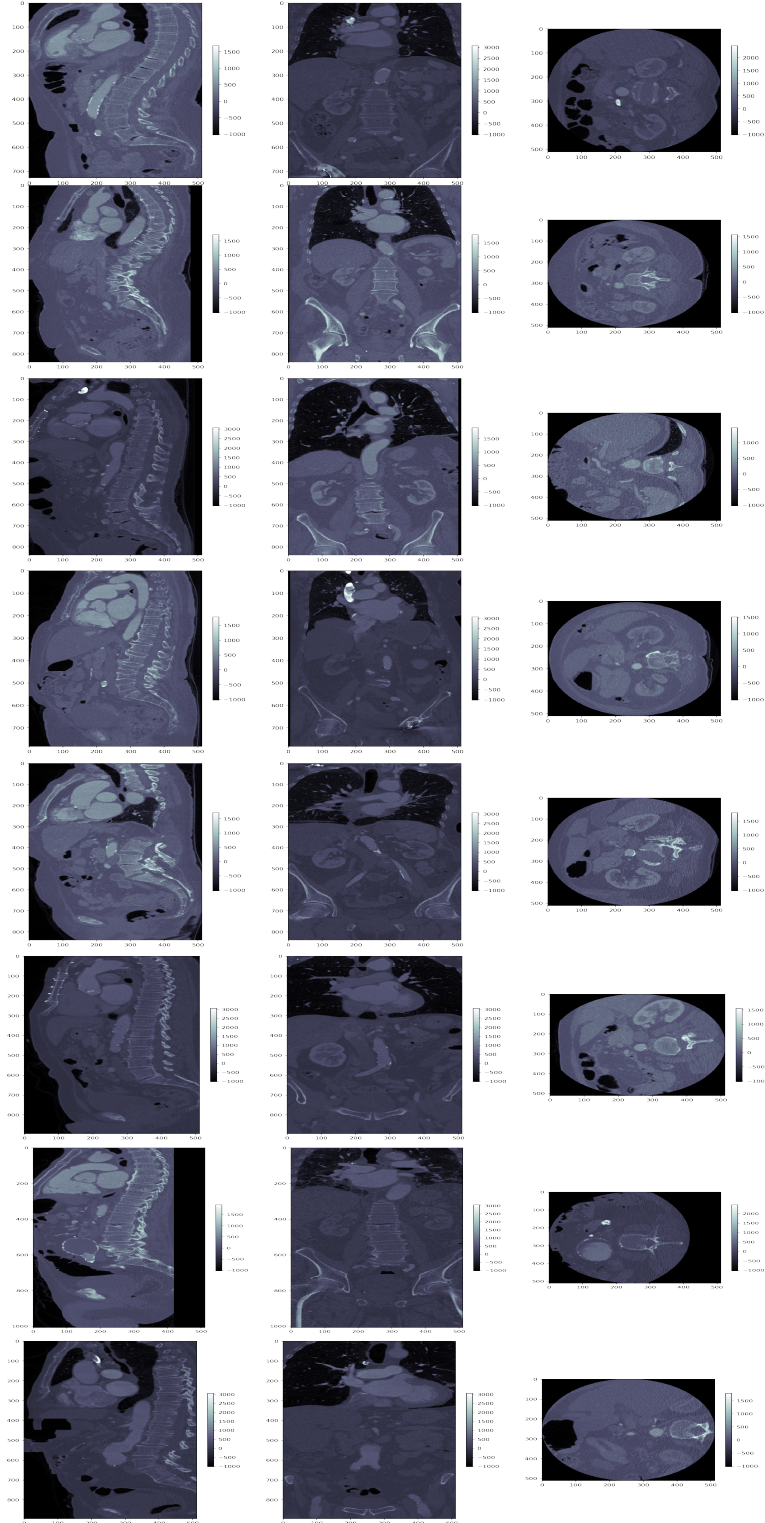


Figure 3.2: Visualisations of three orthogonal slice views of some (16 bit-depths) sample volumes belong to dataset 1.

Volume ID	Pixel Spacing	Slice Thickness	Resolution	Frames	Min. Value	Max. Value
1	[0.488, 0.488]	0.625	512 × 512	728	-1024	3071
2	[0.625, 0.625]	0.625	512 × 512	840	-1024	3071
4	[0.625, 0.625]	0.625	512 × 512	784	-1024	3071
5	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
6	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
7	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
9	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
10	[0.625, 0.625]	0.625	512 × 512	952	-1024	3071
11	[0.625, 0.625]	0.625	512 × 512	952	-1024	3071
13	[0.578, 0.578]	0.625	512 × 512	784	-1024	3071
15	[0.625, 0.625]	0.625	512 × 512	952	-1024	3071
17	[0.625, 0.625]	0.625	512 × 512	1120	-1024	3071
18	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
19	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
20	[0.615, 0.615]	0.625	512 × 512	784	-1024	3071
21	[0.578, 0.578]	0.625	512 × 512	784	-1024	3071
22	[0.625, 0.625]	0.625	512 × 512	840	-1024	3071
23	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
25	[0.625, 0.625]	0.625	512 × 512	896	-1024	3071
26	[0.578, 0.578]	0.625	512 × 512	728	-1024	3071
27	[0.625, 0.625]	0.625	512 × 512	1008	-1024	3071
28	[0.551, 0.551]	0.625	512 × 512	728	-1024	3071
29	[0.488, 0.488]	0.625	512 × 512	896	-1024	3071
30	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
31	[0.703, 0.703]	0.625	512 × 512	840	-1024	3071
32	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
33	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
34	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
35	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
36	[0.580, 0.580]	0.625	512 × 512	896	-1024	3071
38	[0.488, 0.488]	0.625	512 × 512	896	-1024	3071
39	[0.625, 0.625]	0.625	512 × 512	784	-1024	2274
40	[0.625, 0.625]	0.625	512 × 512	782	-1024	3071
41	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
42	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
43	[0.488, 0.488]	0.625	512 × 512	896	-1024	3071
44	[0.625, 0.625]	0.625	512 × 512	1120	-1024	3071
45	[0.429, 0.429]	0.625	512 × 512	1008	-1024	3071
46	[0.488, 0.488]	0.625	512 × 512	840	-1024	3071
47	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
49	[0.488, 0.488]	0.625	512 × 512	952	-1024	3071
50	[0.553, 0.553]	0.625	512 × 512	840	-1024	3071
51	[0.488, 0.488]	0.625	512 × 512	784	-1024	3071

Table 3.1: Overview of dataset 1 details composed of (16 bit-depths) CT medical images.

3.1.2 Dataset 2 - Public CT

This publicly available dataset contains only two volumes from The Cancer Imaging Archive (TCIA) [14,15]. The DICOM files store CT scans of a human lung for patients who suffer from non-small cell lung cancer. The datasets characteristics are presented in table 3.2. An overview of the histogram of voxel intensity across the two volumes (16 bit-depths) belonging to this set is illustrated in Fig. 3.3. This set was only used to evaluate the models in chapter 4.

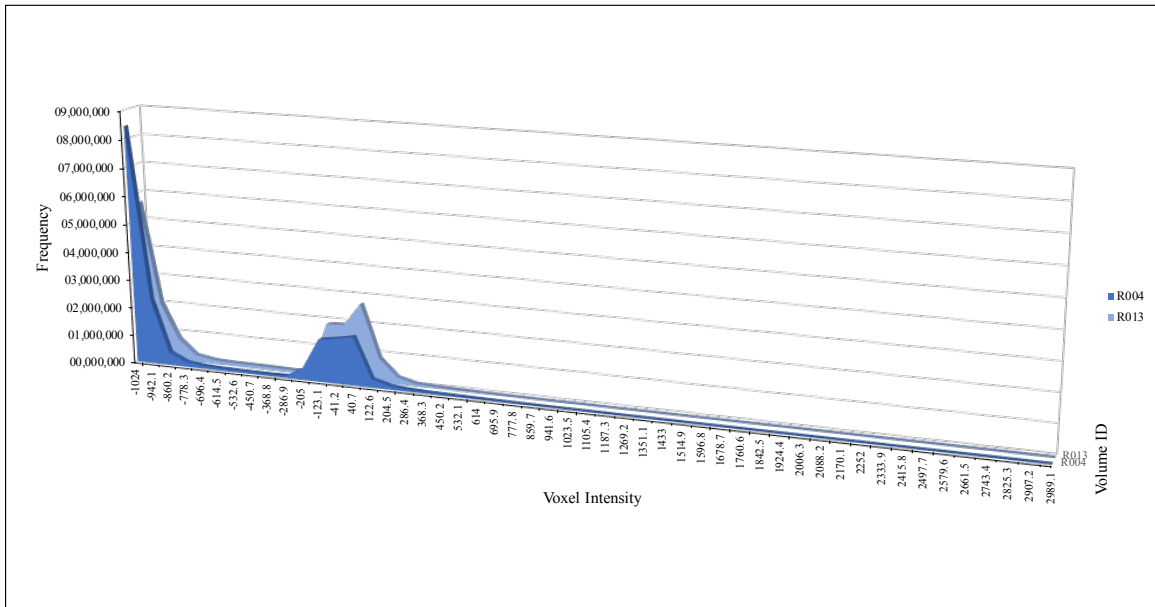


Figure 3.3: Overview of the histogram of voxel intensity in dataset 2 (16 bit-depths).

Volume ID	Pixel Spacing	Slice Thickness	Resolution	Frames	Min. Value	Max. Value
CT_Lung_R004	[0.830078, 0.830078]	5.0	512 x 512	68	-1024	3071
CT_Lung_R013	[0.623047, 0.623047]	5.0	512 x 512	67	-1024	3071

Table 3.2: Overview of dataset 2 information composed of 16 bit-depth CT medical images [14, 15].

3.1.3 Dataset 3 - Public MRI

This set is a public dataset that contains 12 MRI volumes of patients’ head and neck scans stored as (16 bit-depths) grayscale images [16–18]. All volumes have $512 \times 512 \times 120$ resolution, slice thickness 2.0 mm, and $PS = 0.5$ pixel spacing. The minimum intensity value across all volumes is 0, and the maximum is 689. A detailed overview of dataset 3 characteristics is presented in table 3.3. Moreover, visualisations of some sample volumes (16 bit-depths) belonging to this set are presented in Fig. 3.4. This dataset was used as an evaluation set in chapter 5, 6, and 7. Also, it was utilised as a training set only in chapter 7.

Volume ID	Pixel Spacing	Slice Thickness	Resolution	Frames	Min. Value	Max. Value
1	[.5, .5]	2.0	512 x 512	120	0	421
2	[.5, .5]	2.0	512 x 512	120	0	479
3	[.5, .5]	2.0	512 x 512	120	0	494
4	[.5, .5]	2.0	512 x 512	120	0	394
5	[.5, .5]	2.0	512 x 512	120	0	498
6	[.5, .5]	2.0	512 x 512	120	0	527
7	[.5, .5]	2.0	512 x 512	120	0	689
8	[.5, .5]	2.0	512 x 512	120	0	523
9	[.5, .5]	2.0	512 x 512	120	0	452
10	[.5, .5]	2.0	512 x 512	120	0	446
11	[.5, .5]	2.0	512 x 512	120	0	639
12	[.5, .5]	2.0	512 x 512	120	0	406

Table 3.3: Overview of dataset 3 composed of 12 MRI volumes of patients’ head and neck scans (16 bit-depths) [16–18].

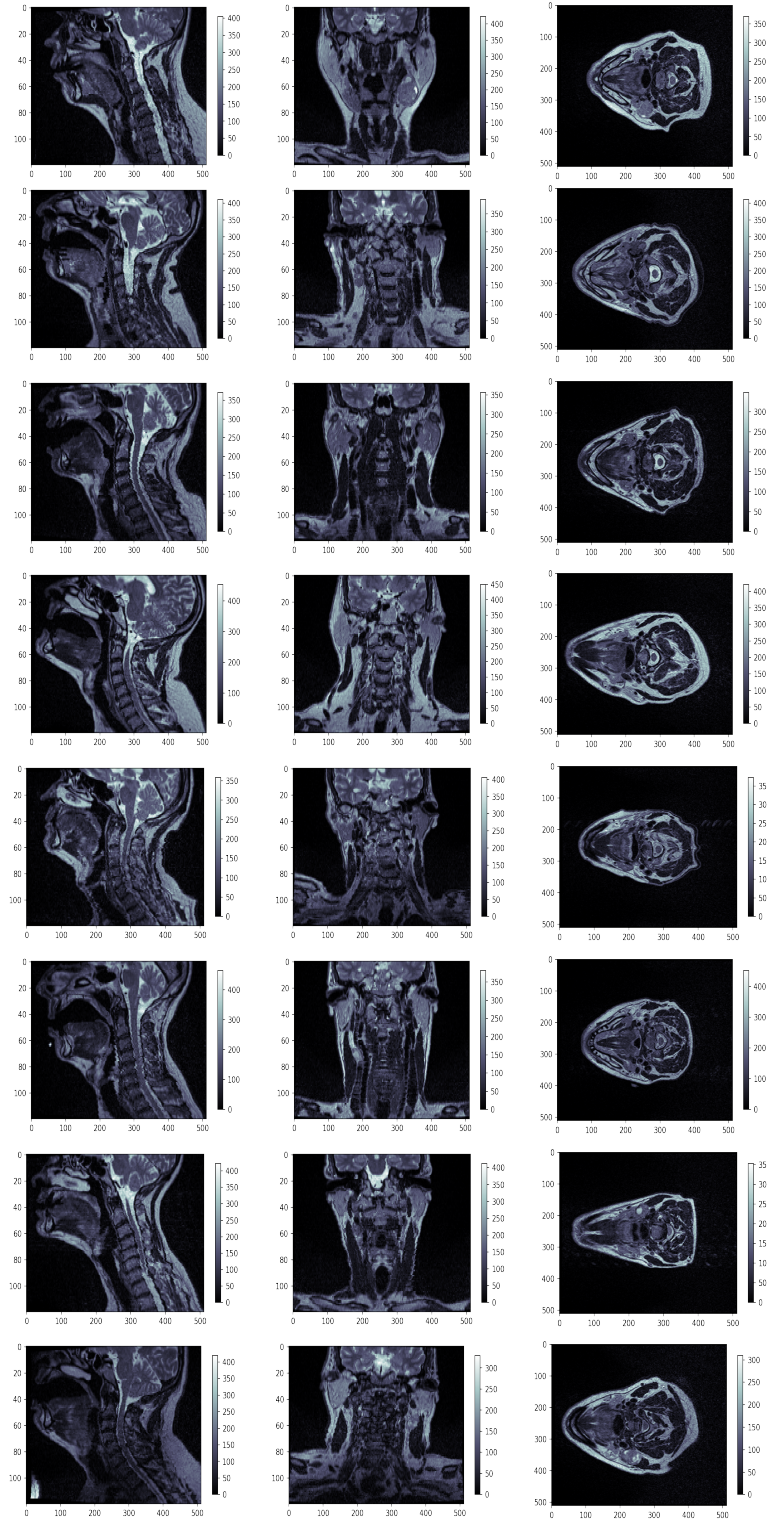


Figure 3.4: Visualisations of three orthogonal slice views of some (16 bit-depths) sample volumes belong to dataset 3.

3.2 Loss Function

Inspired by [150], a joint loss L_{joint} (Eq. 3.1) was chosen as the loss function for all the proposed models proposed in this thesis. This loss function jointly combines the Mean Absolute Error (MAE) (Eq. 3.2) with the Pearson Correlation Coefficient (PCC) (Eq. 3.3) also known as bivariate correlation:

$$L_{Joint} = MAE + \lambda(1 - |PCC|) \quad (3.1)$$

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (3.2)$$

$$PCC = \frac{cov(y_i, \hat{y}_i)}{\sigma_{y_i} \sigma_{\hat{y}_i}} \quad (3.3)$$

Where, y_i is the ground truth voxel value, \hat{y}_i is the models' prediction, n is the total number of data samples, cov is the covariance, σ_{y_i} is the standard deviation of y_i , and $\sigma_{\hat{y}_i}$ is the standard deviation of \hat{y}_i . The intention of using PCC in the loss function is to measure the statistical relationship between the ground truth value y_i and \hat{y}_i . When $PCC = 0$, this means no linear correlation between the two continuous variables. In the L_{joint} , the absolute value of PCC is computed, which limits the PCC value to be less than or equal to 1. If the value is equal to 1, it means the variables are linearly correlated. Incorporating PCC with the MAE for solving the regression problem has a significant impact in enhancing the accuracy and stabilizing the training.

3.3 Evaluation Metrics

3.3.1 Compression Ratio

The compression ratio in bit-per-pixel (bpp) was applied as the primary evaluation criteria across all the technical chapters to compare the compression performance of our proposed codecs to the state-of-the-art lossless codecs. This evaluation metric can be calculated as follows:

$$bpp = \frac{\text{Compressed Image Size (Bits)}}{\text{Number Of Voxels}} \quad (3.4)$$

3.3.2 Compression Time

Compression time is an evaluation metric that measures a codec compression speed by computing its encoding or decoding time (in seconds) – used only in chapters 5 and 6.

Lossless Compression for Volumetric Medical Images Using Deep Neural Network with Local Sampling



Contents

4.1	Introduction	124
4.2	Proposed Method	126
	4.2.1 Local Sampling	127
	4.2.2 Transforming Function	129
4.3	Results and Discussion	129
	4.3.1 Dataset Details	129
	4.3.2 Comparisons with the state-of-the-art	130
4.4	Conclusion and Further Work	134

This work was originally published in the IEEE International Conference on Image Processing 2020, **Awarded ICIP 2020 Top Viewed Q&A Paper Award (2nd place)**, [43], by the thesis author alongside Dr J. Whittle, Dr J. Deng, Prof. B. Mora, and Prof. M. W. Jones.

4.1 Introduction

Data compression forms a central role in handling the bottleneck of data storage, transmission and processing. Lossless compression requires reducing the file size whilst maintaining bit-perfect decompression, which is the main target in medical applications. This chapter presents a novel lossless compression method for 16-bit medical imaging volumes. The aim is to train a neural network (NN) as a 3D data predictor, which minimizes the differences with the original data values and to compress those residuals using arithmetic coding. We evaluate the compression performance of our proposed models to state-of-the-art lossless compression methods, which shows that our approach accomplishes a higher compression ratio compared to JPEG-LS, JPEG2000, JP3D, and HEVC while generalising well.

Medical imaging is used for clinical diagnosis. Precise medical imaging techniques have been developed where radiologists can acquire high quality and high-resolution scans for clinical purposes. 3D medical imaging is often used for further diagnosis and precise pre-surgery planning. According to Diagnostic Imaging Dataset Statistical Release published by NHS, between September 2018 to September 2019, over 45 million medical images were acquired for clinical use, including 5.8M CT scans and 3.7M MRI scans. Data storage for a large amount of medical images poses a great challenge. Most modern hospitals' scanners produce massive volumetric scans with higher resolutions, bit-depths, and vast amounts of data. Consequently, it becomes both demanding and challenging to encode medical images with the guarantee of keeping quality. Especially for clinical purposes, artefacts introduced by lossy compression could result in misleading diagnoses and unfavourable treatment [30].

Lossless compression standards are classified into two main types – image encoders and volumetric encoders. Standard image encoders include JPEG2000 [142], Lossless and Near-Lossless Compression of Continuous-Tone Still Images (JPEG-LS) [13], Context-based Adaptive Lossless Image Codec (CALIC) [134], and Minimum Rate Predictor (MRP) [135]. In contrast to 2D image encoders, volumetric encoders enhance the compression ratio by applying a reduction in a higher-dimensional context, such as: JPEG2000 Part 10 Extensions (JP3D) [7], High-Efficiency Video Coding (HEVC) [9],

3D-CALIC [136], M-CALIC [137], and 3D Minimum Rate Predictor (3D-MRP) [167]. The problem with some of the current lossless compression standards is that they rely on hand-crafted codecs, which may calculate only linear transformations and have limitations in representing non-linear correlations. Recently, state-of-the-art deep neural networks have been demonstrated as feasible to construct both lossy and lossless image compression, which also achieves higher compression ratios than classic linear methods. Thus, the alternative and emerging options involve neural networks to automate the algorithm and introduce non-linearity, which also can converge to a better solution.

State-of-the-art deep learning approaches address lossy reduction to assist purposes such as dimensionality reduction (autoencoders) [168], super-resolution images or video reconstructions [151–153], estimating pixel likelihood (auto-regressive) [148, 155], and generative compression [169, 170]. Less attention has been made to address the lossless performance using NNs. The current deep learning literature for lossless compression usually combine a density estimator model with an arithmetic coder or Asymmetric Numeral System (ANS). The density estimator can be categorised into various types, namely fully connected NN [171], Recurrent Neural Network (LSTM/GRU) DeepZip [145], recursive bits-back coding Bit-Swap [144], and hierarchical probabilistic model L3C [32]. Our proposed model for lossless compression has a fully connected deep NN as its data estimator, which should be faster to train than a Recurrent Network in DeepZip and estimate higher dimensional data (i.e. 3D medical images) compared to [171], which compresses only 2D scans. Additionally, our proposed 3D sampling scheme allows the model to generalise well to unseen samples. **A more in-depth scope of the relevant and recent related works was presented in section 2.4.5.**

Our main contributions in this chapter are:

- A novel 3D predictor model using a neural network that achieves lossless compression for volumetric medical images.
- A computationally efficient model that achieves a higher compression ratio when compared to state-of-the-art lossless compression methods.
- Demonstrate the robustness and generalization of our proposed models experimentally on many datasets for higher dynamic range (16 bit-depths).

The rest of the chapter is organised as follows: In Section 4.2, we introduce the proposed method with a detailed overview of the compression framework. The same

Section 4.2.1 also highlights the 3D local sampling utilised to exploit the spatial similarity and redundancy in a volumetric medical context. The NN model’s architecture and training hyperparameters for our 3D predictor are then described in Subsection 4.2.2. The following Section 4.3, proposes results and discussion for evaluating models’ performance across two distinct datasets. The last Section 4.4, concludes the main findings while highlighting limitations and potentials for development.

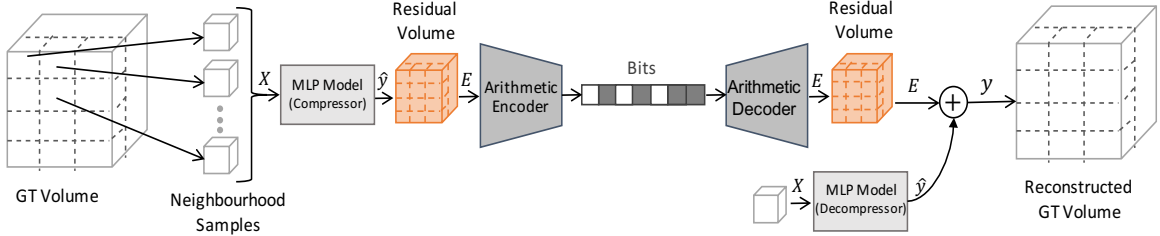


Figure 4.1: An overview of the proposed lossless compression method.

4.2 Proposed Method

We propose (Fig. 4.1) a lossless compression approach using a neural network specifically for volumetric medical images, where the data compression is formulated as a sequential prediction problem. Our approach consists of: the data prediction module called the compressor, the entropy coding using arithmetic coder, and the data recovery module, namely decompressor. Both compressor and decompressor use one neural network model where the same architectures and weights are shared. The model learns the projection function to predict the target voxel given a sequence of samples from its neighbourhood (Fig. 4.2). In order to achieve a high compression ratio for the arithmetic encoder, we train the neural network to minimize the difference between the prediction and the ground-truth (see subsection 2.4.6 for more details on the sequence prediction coding procedure). The regression problem can be solved by learning a mapping function f that predict the output y from an input sequence X through the back-propagation process given a training dataset. The hypothesis is that the prediction is highly correlated with its neighbours’ local appearance and geometric structure. Formally, given a data distribution defined over $X \in R^N$, where X contains input samples from the same distribution (e.g. $X = \{x_1, x_2, \dots, x_n\}$ forms a 1D vector of spatial neighboring voxel-intensities), we learn a differentiable mapping function $\hat{y} = f(X)$ that maps the vector X to a predicted value \hat{y} to minimize the differences with the ground truth voxel value y , where $f(X)$ is represented using a neural network model. Therefore, the residual or prediction error E is calculated as

follows:

$$E = y - \hat{y} \quad (4.1)$$

The residual error E is then encoded using an arithmetic coder and transmitted in a lower bit-rate. The better the model performs in approximating the data distribution, the smaller the residual gets and the lower code-length the coder produces. If many of the error values are negligible (equal zero), this indicates that the model is accurate. All prediction errors are compressed using entropy coding such as Huffman and arithmetic coding to reduce the coding redundancy. Storing and transmitting the compressed error reduces the file size compared to the GT volume. To fully recover the original data from the compressed representation, the bit-stream is decompressed first, and then the residual values E are added to the prediction values \hat{y} (Fig. 4.1). It is important to understand that we compare the compression size of the **volumetric residual/prediction error** (generated by our proposed learning-based models) to the compressed size of the **GT medical volume** (generated by state-of-the-art lossless compression methods).

4.2.1 Local Sampling

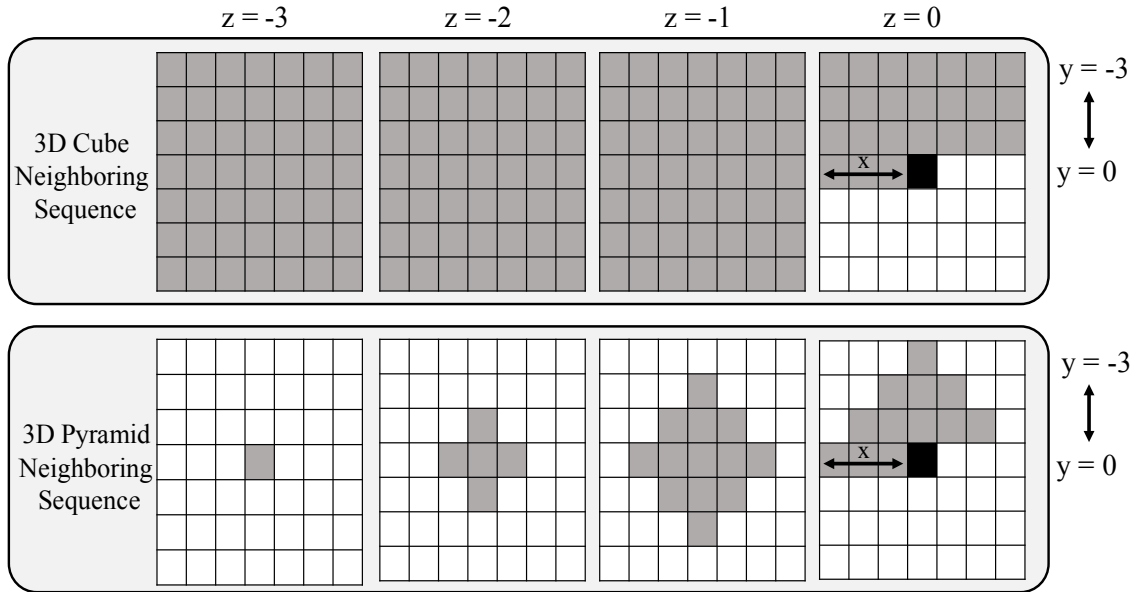


Figure 4.2: Two neighbourhoods used for prediction. $z=0$ represents the current slice, the target voxel for prediction is black, grey voxels are used in the input sequence.

The input sequence forms a crucial role in learning the mapping function of the data distribution. We experimented with differing sampling neighbourhoods and sampling strategies in order to find optimal compression. There is a trade-off between the amount of information presented to the model and the computation cost. Many image-based codecs use the immediate neighbouring pixels (e.g. four previous pixels) to predict the current pixel location. In 3D, neighbours from previous slices are included. These neighbouring sequences allow the encoder to exploit the spatial correlation and redundancy in the 2D plane as well as within the inter-frame region.

Training a model on samples uniformly selected across multiple volumes is problematic, as such a high dynamic searching space does not allow the model to accomplish a maximum compression performance. Therefore, in our proposed chapter, we reduce the sampling space to one volume with the assumption that the data distribution of the human body across the volumes would share some structural similarity and common feature representation since all scans share the same resolution across width and height and are captured by the same hospital with similar scanning parameters (for training set and test set 1). Instead of sampling randomly from the whole volume (uniform voxel sampling) or biasing our sampling towards part of it (3D Gaussian voxel sampling), we extract multiple complete 2D slices across the volume z-axis with a fixed stride (i.e. ten slices from one volume). Then, for each voxel in the ten slices, we extract the 3D neighbouring voxels. We introduce two different shapes of the neighbouring blocks; namely, 3D cube neighbouring sequence and 3D pyramid neighbouring sequence (See Fig. 4.2). In both types, the sequences never include voxels from future slices. All volume values are normalized to the range $[-1, 1]$, and the volume is padded, as determined by the block size, by its minimum voxel value. Padding the volume is crucial in order to include the edge and corner cases in training. All the 3D sequences will be flattened to 1D vectors and randomly shuffled before inputting them to the predictor models. The reason for selecting such sampling schemes is to provide better coverage of the data to benefit the compression ratio. We find that training the model on whole slices with every voxel within the selected slices being available during training leads to an improvement in the final compression ratio on all slices, including those not trained on.

4.2.2 Transforming Function

Layer	Number of Neurons	Activation Function Used
Fully Connected	1024	LeakyReLU
Fully Connected	512	LeakyReLU
Fully Connected	256	LeakyReLU
Fully Connected	128	LeakyReLU
Output	1	Linear

Table 4.1: The proposed neural network architecture.

Multi-layer Perception (MLP) is used to build the sequence prediction models, which consists of an input layer, 4 fully connected hidden layers with non-linear activation functions and followed by a linear output layer as output. The parameter settings of individual layers are given in Table 4.1. A detailed overview of the loss function L_{Joint} (Eq. 3.1) used to train the neural network is given in section 3.2.

Model ID	Sampling Space	Shapes of the input Neighbouring Block	Hyper Parameters
1	All samples were generated from 10 slices extracted from one volume (patient 40)	3D Cube (11x11x11) input sequence	Batch size = 256, learning rate = 0.0002, no L2 regularization, no dropout and no batch normalization
2	All samples were generated from 10 slices extracted from one volume (patient 40)	3D pyramid input sequence (13x13, 9x9, 5x5, 1x1)	Batch size = 32, learning rate = 3e-5, no L2 regularization, no dropout and no batch normalization

Table 4.2: Illustrating the neural network training specifications for the two proposed models.

4.3 Results and Discussion

4.3.1 Dataset Details

Both training set and testing set 1 are from the same data source – generated by the same hospital and with similar scanning parameters such as the slice thickness and spacing between slices but with some variation in the pixel spacing $\in [0.488, 0.5, 0.635, 0.703]$ (see section 3.1.1 for further details on the dataset specifications). The training set consists of ten slices out of 840 slices extracted from one volume (i.e. volume ID 40) with pixel spacing $[0.625, 0.625]$ and slice thickness 0.625. The two proposed

NN models were evaluated on test set 1, consisting of 42 volumes from the same dataset 3.1.1. On the other hand, test set 2 contains only two volumes from a publicly available dataset provided by The Cancer Imaging Archive (TCIA) [16] (see section 3.1.2 for further details on the dataset specifications). Overview of the histogram of voxel intensity across all volumes (16 bit-depths) belonging to test set 1 (Orange) and test set 2 (Blue) is illustrated in Fig. 4.3. By initially exploring the histogram plot across test set 1 and test set 2 volumes, it is noticeable that there are differences in data distributions between the two datasets (e.g. test set 2 has less sample density than set 1 indicating a partial volume effects problem). Other dissimilarities include differences in tissue types, as test set 2 contains non-small lung cancer while test set 1 does not.

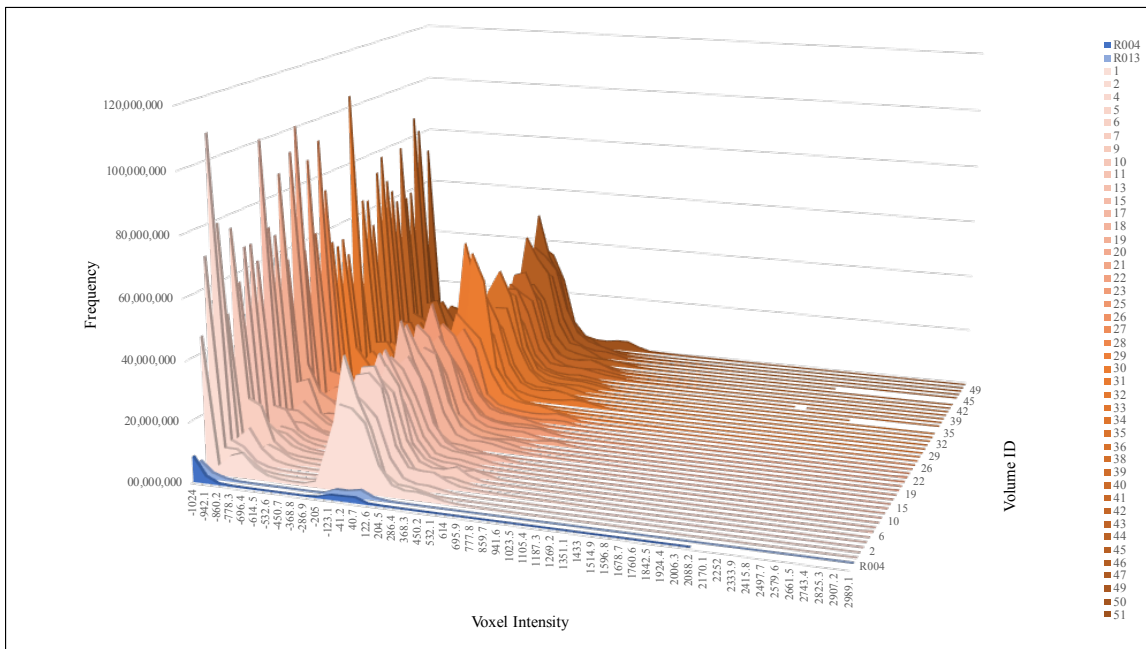


Figure 4.3: Overview of the histogram of voxel intensity across all volumes (16 bit-depths) belonging to the test set 1 (Orange) and test set 2 (Blue).

4.3.2 Comparisons with the state-of-the-art

To select the block size with the best compression performance, we experimentally applied different neighbouring sizes for the 3D cube input sequence, including (3x3x3), (5x5x5), (7x7x7), (9x9x9), and (11x11x11). Figure 4.2 illustrates an example of each input sequence type. In the given examples, $z=0$ represents the current slice, the black voxel refers to the target voxel that needs to be predicted, and the grey voxels are the input sequences while the white voxels are the ones that will be masked because they

are unknown information during decompression. In the given 3D cube example, the block size is set to $(7 \times 7 \times 7)$, implying that the maximum distance from the target voxel in each dimension (x, y, z) is 3. However, in the 3D pyramid neighbouring example, the distance to the target voxel decreases in both x and y axes at each posterior z step. For instance, (at $z=0$, the block size= 7×7), and (at $z=-1$, the block size= 5×5), etc. Based on several experiments, we choose $(11 \times 11 \times 11)$ as the input block size for model 1 as it produces the best compression rate. However, for model 2, the 3D pyramid sequences with $(13 \times 13, 9 \times 9, 5 \times 5, 1 \times 1)$ sequence size are used. The rationale for choosing a pyramid is that it reduces the number of voxels in each training sample (and consequently storage size and training time) but still retains the possibility to integrate information from previous slices into the training.

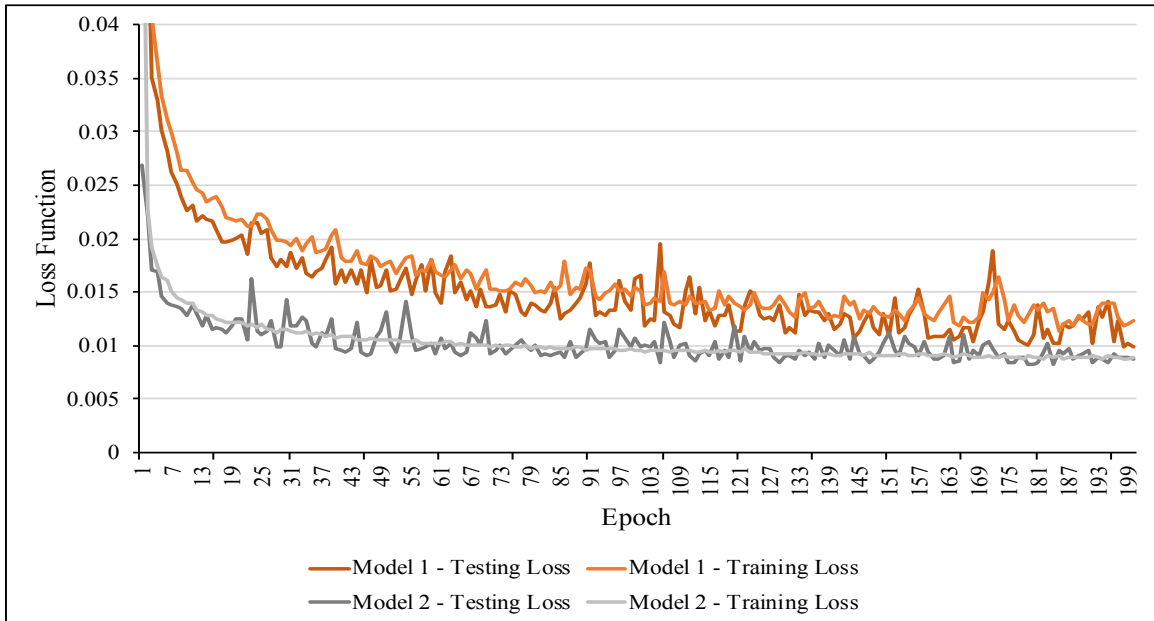


Figure 4.4: An illustration of models loss function plots (i.e. training loss and testing loss) over epochs.

For the neural network, we set the parameters $\lambda = 1$, which weights the contribution of the two losses to be the same. We optimised the neural network using Stochastic Gradient Descent (SGD) with momentum $\beta = 0.9$. Both models have a good stability even without applying batch normalization, weights L2 regularization or dropout. The training hyperparameters of the two models are illustrated in table 4.2. The hyperparameters for model 1 are with batch size of 256, learning rate of 0.0002. However, for model 2, the batch size of 32 and learning rate of $3e-5$ are used. The training hyperparameters were empirically selected based on the machine capabilities,

while all these models were trained until convergence. An illustration of models loss function plots (i.e. training loss and testing loss) over epochs is presented in Fig. 4.4.

We evaluated the compression performance in bpp (Eq. 3.4) of the proposed neural network models in comparison to the state-of-the-art lossless compression methods, including JPEG-LS, JPEG2000, JP3D and HEVC (HM-SCC-extensions-4998) using the lossless configuration with main-RExt profile available in [172]. Figure 4.5, illustrates the compression rate in bpp on test set 1 compressed by the state-of-the-art lossless methods and our two proposed neural network models. The results indicate that the proposed predictor models achieve the best compression ratio on test set 1 compared to the existing methods. Additionally, it is clear that the methods using 3D contents (i.e. two proposed models and JP3D) gained a smaller bit-rate than the ones using 2D content only (i.e. JPEG2000 and JPEG-LS). Among the different compression approaches, our 3D data predictor model (i.e. Model 2) achieves the best compression ratio.

We also evaluated the generalization ability of our models on a completely different data distribution (CT of lung cancer). Our models were not trained on these volumetric medical images. However, it can achieve a close compression ratio to other methods, as shown in table 4.3. This is achieved even though the scanning parameters of the test set 2 differs totally from the training set and test set 1. The slice thickness is 5mm while in training set and test set 1, the thickness is .625 mm, which influences the 3D neighbouring quality learned by our models. Since our neural network models were trained to learn the explicit relation of neighbouring voxels in a specific resolution (i.e. training set), different volume resolution (i.e. test set 2) will influence the performance of the models. However, our model is generalized to gain better compression if the data provided is consistent and has similar structure to the training set (i.e. test set 1 has similar slice thickness but variation in pixel spacing). We tested this by resampling test set 2 to matching characteristics (i.e. pixel spacing of 0.625 with higher sampling density) and compressing it. Our models accomplish the best bit reduction on the new resampled test set compared to the other lossless codecs. Such compression gains are expected as volumes with similar 3D local structures allow our 3D predictors to exploit spatial redundancies more efficiently. Compared to other deep learning approaches such as DeepZip [145], our encoding procedure is fully parallelized and rapid. Decoding time can be enhanced by decoding slices diagonally.

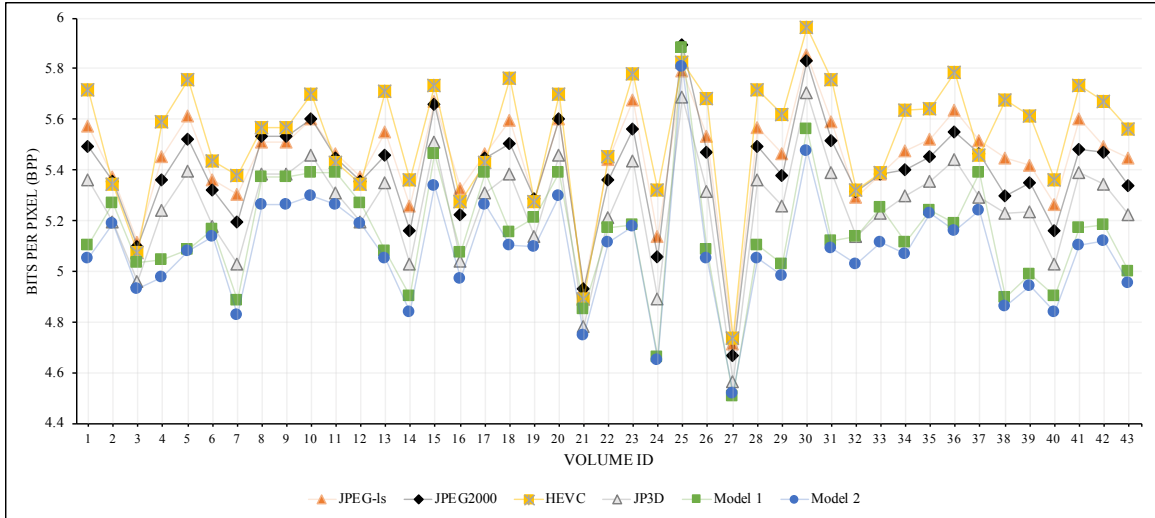


Figure 4.5: Comparing the compression ratio in bpp for the proposed models with the state-of-the-art lossless compression methods over 16-bits volumes on test set 1 (see table 4.3). Another version of this figure that illustrates a scatter plot of the bpp across the same test set 1 ordered by pixel spacing, is presented in B.1

Figure 4.5 and table 4.3 illustrate the compression performance in bpp for all datasets computed by our two proposed models and the existing state-of-the-art lossless methods for 16-bits volumes. The table also includes the volumes scanning parameters including pixel spacing and slice thickness.

Set Type	Volume ID	Pixel Spacing, Slice Thickness	JPEG-ls	JPEG2000	HEVC	JP3D	Model 1	Model 2	
Training Set	40	0.625, 0.625, 0.625	5.387	5.387	5.389	5.23	5.256	5.119	
Test Set 1	1	0.488, 0.488, 0.625	5.571	5.495	5.715	5.362	5.106	5.056	
	2	0.625, 0.625, 0.625	5.374	5.357	5.346	5.196	5.272	5.189	
	4	0.625, 0.625, 0.625	5.117	5.1	5.077	4.961	5.036	4.934	
	5	0.488, 0.488, 0.625	5.455	5.361	5.589	5.243	5.045	4.982	
	6	0.488, 0.488, 0.625	5.616	5.525	5.755	5.396	5.085	5.084	
	7	0.488, 0.488, 0.625	5.359	5.32	5.437	5.18	5.167	5.138	
	9	0.488, 0.488, 0.625	5.306	5.197	5.378	5.033	4.887	4.83	
	10	0.625, 0.625, 0.625	5.511	5.534	5.566	5.385	5.375	5.265	
	11	0.625, 0.625, 0.625	5.511	5.534	5.566	5.385	5.375	5.265	
	13	0.578, 0.578, 0.625	5.603	5.605	5.701	5.461	5.393	5.302	
	15	0.625, 0.625, 0.625	5.463	5.448	5.429	5.311	5.388	5.266	
	17	0.625, 0.625, 0.625	5.373	5.357	5.346	5.196	5.272	5.189	
	18	0.488, 0.488, 0.625	5.552	5.46	5.711	5.353	5.085	5.056	
	19	0.488, 0.488, 0.625	5.262	5.162	5.36	5.033	4.905	4.841	
	20	0.615, 0.615, 0.625	5.669	5.66	5.737	5.508	5.464	5.342	
	21	0.578, 0.578, 0.625	5.329	5.228	5.275	5.04	5.077	4.973	
	22	0.625, 0.625, 0.625	5.465	5.448	5.429	5.311	5.388	5.266	
	23	0.488, 0.488, 0.625	5.599	5.507	5.761	5.386	5.155	5.105	
	25	0.625, 0.625, 0.625	5.283	5.286	5.274	5.137	5.214	5.101	
	26	0.578, 0.578, 0.625	5.605	5.605	5.701	5.461	5.393	5.302	
	27	0.625, 0.625, 0.625	4.935	4.931	4.895	4.786	4.854	4.752	
	28	0.551, 0.551, 0.625	5.444	5.363	5.454	5.213	5.174	5.117	
	29	0.488, 0.488, 0.625	5.676	5.562	5.779	5.437	5.186	5.18	
	30	0.488, 0.488, 0.625	5.137	5.057	5.321	4.895	4.666	4.653	
	31	0.703, 0.703, 0.625	5.793	5.896	5.827	5.686	5.88	5.807	
	32	0.488, 0.488, 0.625	5.535	5.471	5.682	5.318	5.09	5.053	
	33	0.488, 0.488, 0.625	4.717	4.671	4.738	4.569	4.512	4.524	
	34	0.488, 0.488, 0.625	5.57	5.495	5.715	5.362	5.106	5.056	
	35	0.488, 0.488, 0.625	5.468	5.378	5.618	5.26	5.031	4.988	
	36	0.580, 0.580, 0.625	5.853	5.829	5.963	5.703	5.563	5.477	
	38	0.488, 0.488, 0.625	5.59	5.515	5.756	5.391	5.12	5.093	
	39	0.625, 0.625, 0.625	5.294	5.317	5.322	5.137	5.139	5.031	
	41	0.488, 0.488, 0.625	5.478	5.404	5.638	5.3	5.117	5.073	
	42	0.488, 0.488, 0.625	5.525	5.456	5.641	5.355	5.242	5.23	
	43	0.488, 0.488, 0.625	5.638	5.552	5.788	5.441	5.189	5.163	
	44	0.625, 0.625, 0.625	5.516	5.467	5.461	5.296	5.389	5.242	
	45	0.429, 0.429, 0.625	5.448	5.3	5.675	5.233	4.901	4.868	
	46	0.488, 0.488, 0.625	5.417	5.353	5.615	5.239	4.99	4.945	
	47	0.488, 0.488, 0.625	5.263	5.162	5.36	5.033	4.905	4.841	
	49	0.488, 0.488, 0.625	5.605	5.484	5.736	5.389	5.174	5.106	
	50	0.553, 0.553, 0.625	5.491	5.473	5.674	5.348	5.184	5.122	
	51	0.488, 0.488, 0.625	5.446	5.341	5.563	5.223	5.003	4.959	
	Test Set 2	CT_Lung_R004	0.830, 0.830, 5.000	5.937	6.014	5.739	5.967	6.664	6.715
		CT_Lung_R013	0.623, 0.623, 5.000	5.747	5.539	5.835	5.623	5.959	5.847
	Resampled Test Set 2	CT_Lung_R004	0.625, 0.625, 0.625	5.459	5.243	-	5.195	4.915	4.904
		CT_Lung_R013	0.623, 0.623, 0.625	5.698	5.485	-	5.375	5.237	5.238

Table 4.3: Compression performance in bpp for all datasets computed by our two proposed models and the existing state-of-the-art lossless methods for 16-bits volumes. The best compression result (bpp), is in bold.

4.4 Conclusion and Further Work

In this chapter, we proposed a novel lossless compression system using a neural network for volumetric medical image (16 bit). Two localized sampling methods were introduced and evaluated on real 3D volumetric medical imaging datasets. The comparison

study shows that our method outperforms the standard lossless compression methods. It also suggests that the proposed method is feasible to generalize to unseen dataset while retains satisfactory performance. Further work includes: Study of generalization across samples with different pixel spacing or scan quality. Moreover, it would be interesting to extend the applications of our method to include different image type and modalities (e.g. MRI, and RGB images). The effect of model size and weight sparsity on compression ratio from transmitting both the compressed representation and decoder. Optimization of the decoder to leverage parallelism over the diagonal leading edge to reduce decode time at small batch sizes.

In the following chapter, we consider a further investigation of using a neural network as a 3D data predictor in various directions, including network architecture and input sampling space. We investigate the use of a different Neural Network architecture (i.e. Long Short-Term Memory (LSTM)) as a 3D data predictor to compress volumetric medical images (with 16 bits-depth) losslessly. The study evaluates the compression performance of the proposed models (i.e. MedZip) compared to state-of-the-art lossless compression methods over various medical scans modalities.

MedZip: 3D Medical Images Lossless Compressor Using Recurrent Neural Network (LSTM)



Contents

5.1	Introduction	138
5.2	Proposed Method	140
	5.2.1 Overview	140
	5.2.2 Network Architecture	141
	5.2.3 Local Sampling	141
5.3	Results and Discussion	142
	5.3.1 Dataset	142
	5.3.2 The Proposed Models	142
	5.3.3 Comparisons with the state-of-the-art	145
5.4	Conclusion and Further Work	150

This work was originally published in the 25th International Conference on Pattern Recognition (ICPR) 2020, [44], by the thesis author alongside Dr J. Whittle, Dr J. Deng, Prof. B. Mora, and Prof. M. W. Jones.

5.1 Introduction

Modern hospital imaging and scanning equipment produce massive volumetric scans with higher resolutions, bit-depths, and vast amounts of data. Building on our motivation chapter 1, the number of medical images is increasing yearly, which also demands reducing storage overhead while guaranteeing exact reconstruction of clinical data for accurate diagnosis. Based on our findings in the previous chapter 4, the performance of the learning-based sequence prediction models is mainly influenced by the various pixel spacings and scanning quality. Thus, in this chapter, we seek to improve the compression performance by enhancing the model generalisability across different scanning settings and modalities. We hypothesise that by selecting training samples with different scanning settings while using a recurrent network as our learning-based model, our compression framework would have better memorability of voxel dependencies and hence better generalisability. Among the other RNN-based models, we choose the LSTM model as 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 its gating mechanism. These features allow LSTM to learn long-term dependency with more stabilised training.

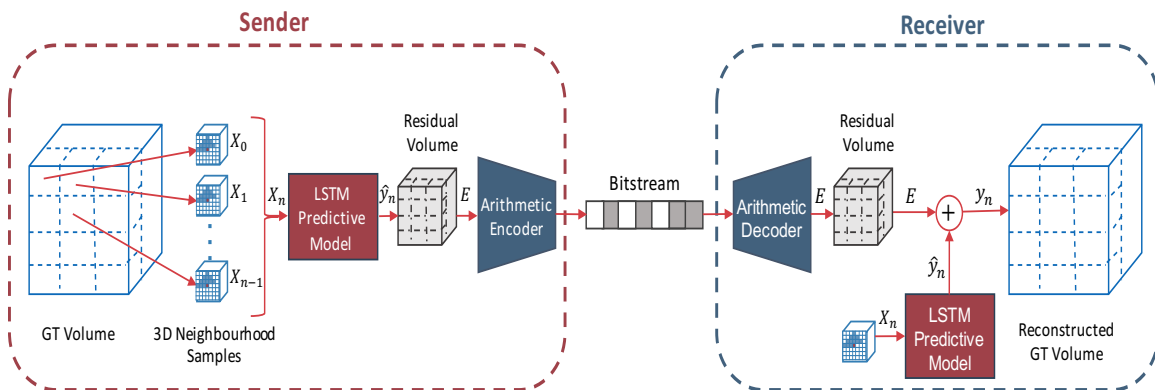


Figure 5.1: An overview of our proposed lossless compression framework using LSTM model.

Compression performance depends on the type of data redundancy, which is classified into three main types, spatial, coding, and spectral (psycho-visual) redund-

ancy [160]. Each type represents the correlation between different parts of the data, and by reducing such repetition, a coder gains compression. For instance, spatial redundancy computes the correlation of a pixel value based on its neighbouring pixels intensities. If such correlation was computed within a 3D space (e.g. between video frames), then it is known as temporal redundancy (interframe coding). On the other hand, computing input statistics to apply variable length coding is known as coding redundancy. In spectral (psycho-visual) redundancy, correlation is computed over visual perception (e.g. colour planes or spectral bands) (covered in details in section 2.3.1). **Recent related works covering the scope of this chapter in-depth was presented in section 2.4.5.**

Deep learning approaches form a promising and emerging research direction due to their ability to estimate non-linear transformations and data likelihood. Current state-of-the-art deep learning methods illustrate that neural networks can construct lossy and lossless performance with results comparable to or even better than standard linear codecs. This chapter proposes a lossless compression approach using LSTM specifically for 16-bit volumetric medical images reduction.

Our proposed lossless compression framework integrates a recurrent neural network (LSTM) to learn spatial correlations of neighbourhood sequences within the 3D regions, followed by an entropy encoding for the residual errors. A high-level overview of our proposed lossless compression framework is given in Fig. 5.1. Given a set of neighbouring voxels, the LSTM model predicts the next intensity value in the sequence. The prediction error is calculated as the difference between the Ground Truth (GT) voxel and the predicted one.

The contributions of this chapter are as follows:

- A novel lossless compression method using LSTM recurrent neural network cells that achieves a higher compression ratio compared to state-of-the-art lossless compression methods.
- Our proposed approach MedZip is for a domain-specific application, namely, volumetric medical images (with 16 bit-depths).
- We demonstrate the generalization of our proposed models on many datasets for a higher dynamic range.

The rest of the chapter is outlined as follows. Section 5.2 presents the compression framework, which contains the LSTM model as a many-to-one sequence prediction model. A detailed description of the network architecture, training datasets, and

training hyperparameters for each of the three proposed MedZip models are given 5.2.3. Experimental results and discussions are introduced in Section 5.3. In Subsection 5.3.2, compression performance (bpp and time) of two different neighbourhood shapes are compared. The same Section 5.3.3 propose details of experimental evaluation of the proposed models with a comparison against the current state of the art lossless compression alternatives. Finally, the main findings and potentials for future work are reported in Section 5.4.

5.2 Proposed Method

5.2.1 Overview

As the LSTM model is one of the state-of-the-art sequence models, we formulated our proposed lossless compression approach as a supervised sequence prediction problem and integrated the LSTM model as a 3D sequence predictor (see Fig. 5.1). Our LSTM model solves a many-to-one sequence prediction problem, which takes a sequence of 3D neighbouring voxels as input and predicts the next intensity value. The intention of using LSTM memory cells to solve the sequence prediction is its ability to maintain the gradient flow across the cells in a way considerably better than RNN. The gates mechanisms in LSTM controls updating the long dependencies with less potential to get gradient vanishing or exploding problems. Such flow control (gates) allows the internal memory of LSTM to learn the long-term inter-frame correlation between slices as well as the spatial correlation. As shown in Fig.5.1, both sender and receiver have the same LSTM predictive model with the same architectures and sharing the same weights. Given a data distribution defined over $V \in R^N$, we extract training sequence samples $X_n \subset V$, where each $X_i = \{x_1, x_2, \dots, x_{l-1}\}$ is a flattened vector containing 3D neighboring intensities of voxel y_i and l is the sequence length of the 3D neighborhood. The LSTM model is trained to learn a differentiable mapping function $\hat{y}_i = f(X_i)$ that maps a sequence input X_i to a single output \hat{y}_i to minimize the difference with y_i (ground truth voxel). After training, the model applies compression, with the sender computing the residual error (prediction error) E for each voxel as:

$$E = y_n - \hat{y}_n \tag{5.1}$$

The error is compressed with arithmetic coding and sent or stored with a lower bit-rate. To reconstruct the original data by the receiver, a reversed operation is applied by the arithmetic decoder, which losslessly reconstructs the residual error E . The LSTM

model is used to generate predictions \hat{y}_n , which are added to the error values E as illustrated in figure 5.1.

5.2.2 Network Architecture

The proposed models are LSTM models, which are composed of the input layer, LSTM layer with 128 cells, and a linear output layer. The activation functions of the LSTM cells are sigmoid and tanh. The parameter settings of each layer is presented in Table 5.1. Details on the loss function used for training all the models are provided in section 3.2.

Layer	Number of Neurons	Activation Function Used
LSTM	128	Sigmoid and Tanh
Output	1	Linear

Table 5.1: The proposed LSTM architecture.

5.2.3 Local Sampling

In a sequence prediction model, the shape and amount of the input sequence form a crucial role in learning the mapping function of the data distribution to the target output. The sequence length is determined by the number of previous observations or features imposed with some explicit order that a model analyses before making a prediction. However, within a recurrent neural network, the challenge is that as the sequence length increases, the model will suffer from the vanishing or exploding gradient problem. Practically, by breaking the input into a smaller fixed sequence length, this problem can be avoided. Image compression generally utilises the four previous neighbouring pixels to encode the target pixel. Video and 3D predictor codecs involve neighbouring pixels from previous frames to discover temporal (inter-frame) redundancy.

In our proposed sequence prediction model, we formulate the input sequence to utilise 3D neighbours (i.e. integrating information from previous slices) with a decreased number of voxels. We refer to this sequence pattern as 3D pyramid (see Fig. 5.2.b). A comparison to block-based models is given in section 4.3. The aim is to learn 3D spatial correlation within the volume space by fetching a fixed-length representative input sequence that can lead to optimal compression. We choose a 3D pyramid sequence with $(13 \times 13, 9 \times 9, 5 \times 5, 1 \times 1)$ regions on each slice to be the input sequence to our models. To choose the training set we sampled uniformly through

multiple volumes with a specific pixel spacing. For each voxel in the set, we extract the 3D pyramid neighbouring sequences (edge voxels are padded with minimum intensity). All volume values are normalized to the range $[-1, 1]$. Each sequence prediction model is trained to make one-step ahead prediction until convergence.

Model ID	Training Set (Pixel Spacing)	Training Set (Slice Thickness)	Hyper Parameters	Preprocessing
MedZip1	Random samples from volumes with pixel spacing .488	.625	Batch size=128, & learning rate=0.00005	-3D pyramid neighboring sequence with $(13 \times 13, 9 \times 9, 5 \times 5, 1 \times 1)$ sequence size. -All samples values are normalized between $[-1, 1]$.
MedZip2	Random samples from volumes with pixel spacing .625	.625	Batch size=128, & learning rate=0.00005	
MedZip3	Random samples from volumes with pixel spacing .488, .578, .625	.625	Batch size=128, & learning rate=0.0001	

Table 5.2: An overview of the training set and the training hyper parameters for each of the proposed LSTM models.

5.3 Results and Discussion

5.3.1 Dataset

The two datasets used in this chapter consist of a set of DICOM files with two different modalities, Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), stored in 16-bit grayscale images. Dataset1 is a private dataset containing 43 volumes representing CT scans for a patient’s entire trunk (for further details on the dataset specifications, refer to section 3.1.1). The proposed three models were trained on random subsets extracted from volumes with a specific pixel spacing (i.e. MedZip1 and MedZip2) or a subset obtained from various pixel spacing (i.e. MedZip3 training set). Table 5.2 presents an overview of the training set for each model. The evaluation was conducted on TestSet1, which contains the rest of the volumes belonging to Dataset1. We also evaluate our three proposed models on a public dataset (TestSet2), which has a different modality (MRI) and represents a different part of the patient’s body, namely, the head and neck [16–18] (see section 3.1.3 for further details on the dataset specifications).

5.3.2 The Proposed Models

This subsection provides the experiential and training details for the three proposed sequence prediction models. For all these models, we used a vanilla LSTM architecture (see section 5.2.2). Since the capacity of the recurrent network is determined by the

number of memory cells the network has, we found that the LSTM with 128 cells has enough capacity to learn the 3D voxel correlation. This relatively compact network reduces the overhead of model size.

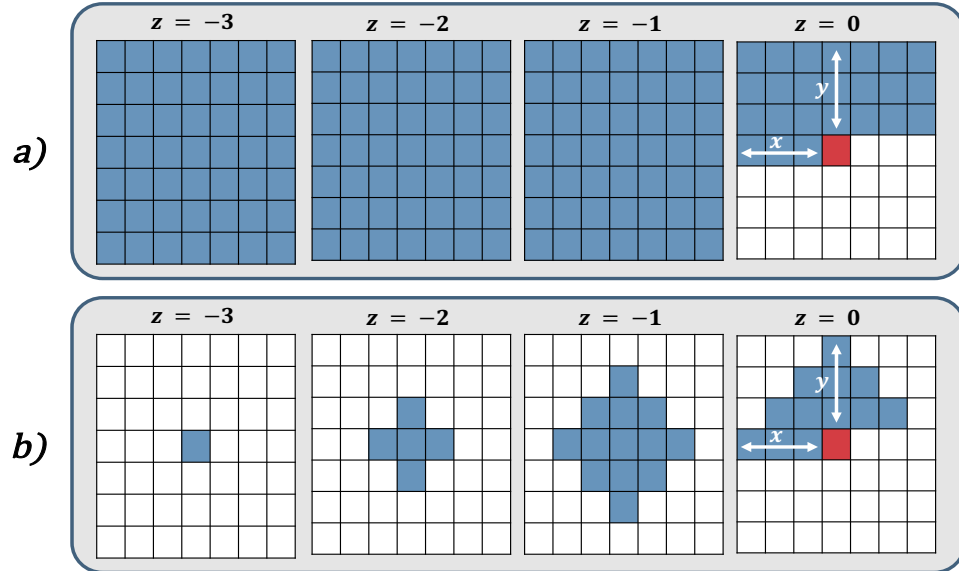


Figure 5.2: Two different neighborhood shapes: a) 3D cube neighboring sequence and b) 3D pyramid neighboring sequence. $z=0$ represents the current slice. The red voxel refers to the target voxel that needs to be predicted, and blue voxels are the input sequences while the white voxels are ignored (masked)

	3D Pyramid Neighboring Sequence	3D Cube Neighboring Sequence		
Neighborhood Block Size	(13x13,9x9, 5x5,1x1)	(5x5x5)	(7x7x7)	(9x9x9)
bits-per-pixel (bpp)	4.267	4.702	4.478	4.36
Compression Time (hh:mm:ss)	1:23:58	0:44:51	1:17:13	2:27:47

Table 5.3: Comparing the compression performance (compression ratio (bpp) and compression time (s)) of different neighboring sequence (3D pyramid & 3D cube) with different block sizes.

Two different 3D neighbourhood shapes were applied to find the input sequence that can lead to an optimal compression, namely, the 3D cube and the 3D pyramid neighbouring sequence. Each type introduces a diverse coverage of the block around the target voxel. Figure 5.2 illustrates an example of each 3D shape. For both types,

$z=0$ represents the current slice. The sequence only includes voxels from the current slice and previous slices. Any future voxels in the three axes (i.e. x , y , and z -axis) are not included as they are unknown during decompression. The red voxel refers to the target voxel that needs to be predicted, and blue voxels are the input sequences while the white voxels are ignored (masked). In the same figure, a) the 3D cube neighbourhood size is $(7 \times 7 \times 7)$ while b) the 3D pyramid neighbourhood size is $(7 \times 7, 5 \times 5, 3 \times 3, 1 \times 1)$.

The intention for choosing the 3D pyramid neighbouring sequence with this specific shape is built upon applying the Manhattan distance from the target voxel to a certain distance in each dimension. The 3D local appearance and structures tend to be highly correlated, leading to spatial and interframe redundancy reductions. Experimentally, different lengths to the target voxel were applied to select the 3D neighbouring size with the best compression performance as presented in Table 5.3. Both compression ratio in (bpp) and compression time have been computed to select the optimal input sequence. As illustrated in Table 5.3, the pyramid structure was compared to a full block neighbourhood at $(5 \times 5 \times 5)$, $(7 \times 7 \times 7)$, and $(9 \times 9 \times 9)$ block sizes. As expected, with the increase in the 3D cube block size, the compression rate also increases as well as the compression time due to the longer the sequence length becomes. The 3D pyramid neighbourhood demonstrates a great balance between the compression time and overall compression achievement. Therefore, the 3D pyramid sequence with $(13 \times 13, 9 \times 9, 5 \times 5, 1 \times 1)$ sequence size was used as input for all the proposed models. We found that compared to using a full cube block, there was no performance loss in terms of the size of the compressed file, and the training time was substantially reduced because fewer samples were used.

The training hyperparameters and training set specifications of the three models are provided in Table 5.2. As Dataset1 has variance in the pixel spacing between volumes, we empirically investigate the effect on compression ratio when training on different spacing. MedZip1 was trained on samples selected randomly from volumes with $[\text{.488}, \text{.488}]$ pixel spacing. MedZip2 was trained on samples from volumes with $[\text{.625}, \text{.625}]$ spacing. MedZip3 was trained on subsets sampled equally from volumes with all different spacing $[\text{.488}, \text{.488}]$, $[\text{.578}, \text{.578}]$ and $[\text{.625}, \text{.625}]$ pixel spacing. Both MedZip1 and MedZip2 were trained with batch size=128 and learning rate=0.00005. However, MedZip3 training hyperparameters are batch size=128 and learning rate=0.0001. All LSTM networks were optimized by Adam optimizer with $\beta_1 = 0.9$, and $\beta_2 = 0.999$ while training until convergence. The parameter λ in the joint loss L_{joint} (equation 3.1) was set to $\lambda = 1$, which weights the contribution of the two losses to be the same.

5.3.3 Comparisons with the state-of-the-art

In this subsection, we evaluate the compression performance of our three proposed models on two different medical images datasets (TestSet1 and TestSet2). We compare the compression size of our models to the state-of-the-art lossless compression methods. The selected methods include image-based codecs, namely, JPEG-LS [173], and JPEG2000 (OpenJPEG software) [174] and 3D volumetric codecs – JP3D [174], and HEVC (HM-SCC-extensions-4998) using lossless configuration with main-RExt profile available in [172] [159]. The evaluation compression also includes the Prediction by Partial Matching (PPMd) algorithm [175] with ultra compression level. Lastly, we have benchmarked our approach against a deep learning lossless compression method known as LSTM-Compress [146], which follows an entropy-based coding methodology (see subsection 2.4.5.2 for further details).

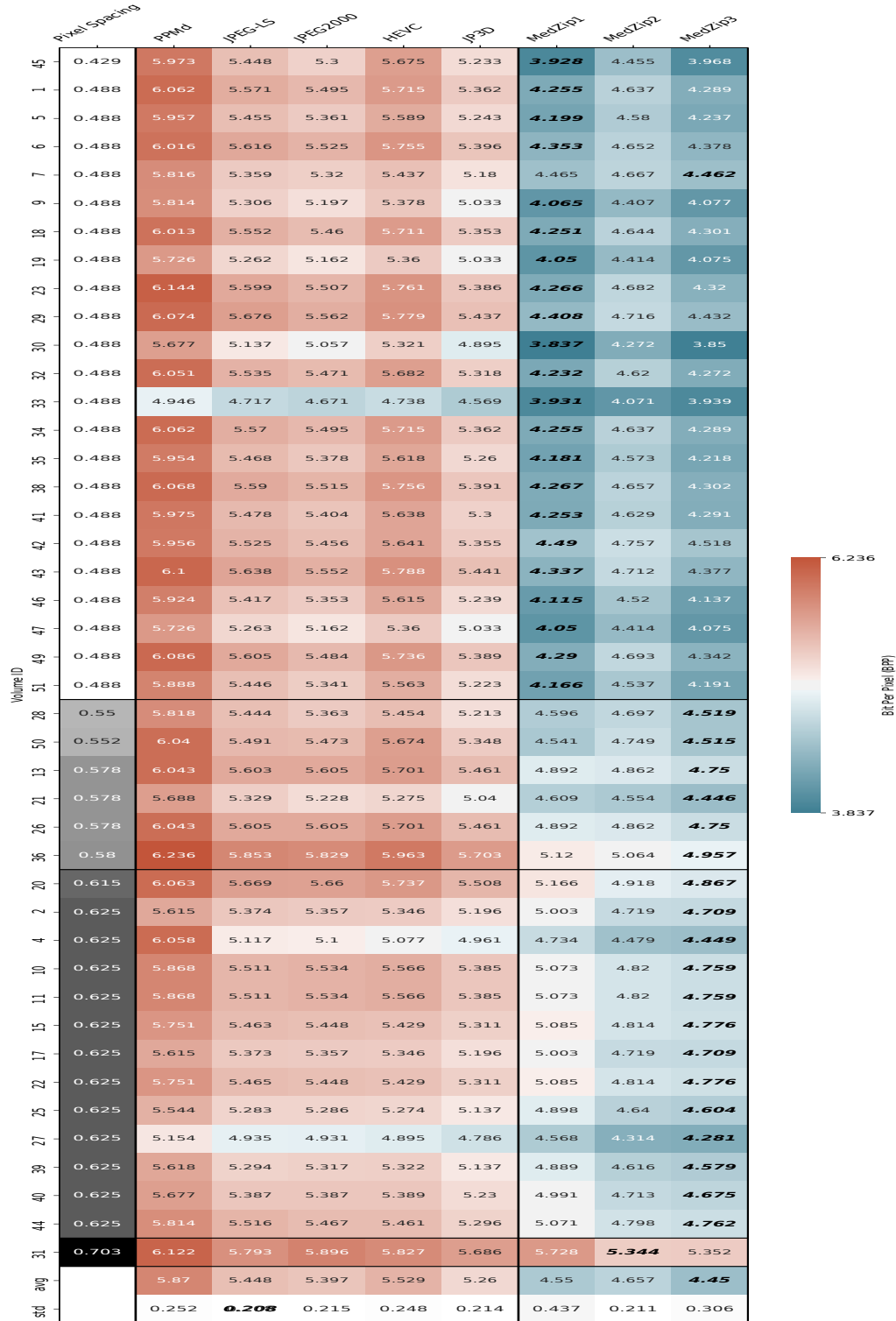


Figure 5.3: The Bits-per-pixel (bpp) for each lossless compression method was measured over TestSet1. The first column is colour mapped by the pixel spacing value of each volume. The other cells are highlighted from the maximum compression of 3.837 bpp (Blue) to the minimum compression of 6.236 bpp (Red). For the average row, MedZip3 saves 15.4% space compared to the best performer (JP3D).

	Pixel Spacing	PPMd	JPEG-LS	JPEG2000	HEVC	JP3D	MedZip1	MedZip2	MedZip3
avg	0.488	5.915	5.445	5.359	5.58	5.237	4.204	4.564	4.234
std	0.488	0.2531	0.2147	0.2085	0.2409	0.2112	0.1715	0.1657	0.1749
avg	0.578	5.965	5.539	5.492	5.619	5.35	4.688	4.761	4.59
std	0.578	0.1805	0.1689	0.2036	0.2174	0.2161	0.3106	0.1868	0.2505
avg	0.625	5.723	5.377	5.371	5.372	5.218	4.972	4.706	4.67
std	0.625	0.2366	0.1903	0.1905	0.2143	0.1893	0.1653	0.1635	0.1585
avg overall		5.87	5.448	5.397	5.529	5.26	4.55	4.657	4.45
std overall		0.2519	0.208	0.2155	0.2477	0.2138	0.4365	0.2113	0.3056

Figure 5.4: An illustration of the bpp average and the standard deviation over TestSet1 groups for all the lossless compression methods from Fig. 5.3. The first column is colour mapped by the pixel spacing value of each set in TestSet1.

TestSet1. The evaluation over TestSet1 is illustrated in Fig.5.3. The volumes are classified into four sets based on their pixel spacing value, separated by horizontal black lines. Each row represents the bpp over a single volume except for the last row, which computes the average of (bpp) for each method through all volumes. The first column presents the pixel spacing of each volume in Dataset1. Cells in the first column are highlighted from minimum pixel spacing .488 (White) to maximum spacing .703 (Black). Each other column represents the bits per pixel (bpp) of a lossless compression method, including PPMd, JPEG-LS, JPEG2000, HEVC, JP3D, MedZip1, MedZip2, and MedZip3, respectively. Cells are highlighted from the maximum compression 3.837 bpp (Blue) to the lowest compression 6.236 bpp (Red). Overall, the performance of our RNN models illustrates the best improvement in compression compared to the standard algorithms. Among the standard codecs, JP3D has a better bit-per-pixel than the image-based codecs (i.e. JPEG-LS and JPEG2000) by exploiting of 3D correlation within frames. HEVC 3D-codec gave worse than average compression results on volumes with pixel spacing .488 and .57 but slightly better performance than image-codecs (JPEG-LS and JPEG2000) on volumes with larger

pixel spacing. For the standard compression methods, PPMd produces the least compression ratio overall volumes in TestSet1. Of our new methods, MedZip1 has the best compression ratio for volumes with .488 pixel spacing – the spacing it was trained-on. Less performance was gained on volumes with larger pixel spacing but still better than classical approaches. MedZip2, which was trained on .625 pixel spacing, gains a considerably good compression ratio overall regardless of volumes pixel spacing. MedZip3 has a good compression performance overall on volumes with various pixel spacing demonstrating generalization. The robustness of this model was gained by training it on samples that belong to volumes with different pixel spacing.

	Pixel Spacing	PPMd	JPEG-LS	JPEG2000	HEVC	JP3D	LSTM-Compress	MedZip1	MedZip2	MedZip3
1	0.5	4.245	3.941	3.558	4.079	3.707	3.424	3.053	3.267	3.082
2	0.5	4.246	3.966	3.57	4.068	3.702	3.455	3.114	3.294	3.133
3	0.5	4.281	3.969	3.572	4.096	3.709	3.458	3.1	3.299	3.122
4	0.5	4.133	3.875	3.487	3.99	3.621	3.379	3.014	3.212	3.037
5	0.5	4.52	4.158	3.757	4.273	3.885	3.654	3.295	3.478	3.314
6	0.5	4.101	3.827	3.438	3.96	3.579	3.338	2.949	3.172	2.976
7	0.5	4.163	3.899	3.516	4.025	3.662	3.428	3.099	3.279	3.116
8	0.5	4.327	4.04	3.644	4.137	3.776	3.536	3.222	3.382	3.238
9	0.5	4.201	3.922	3.534	4.034	3.679	3.459	3.142	3.304	3.158
10	0.5	4.257	3.969	3.581	4.101	3.719	3.483	3.137	3.32	3.155
11	0.5	4.153	3.879	3.503	4.029	3.655	3.409	3.012	3.236	3.047
12	0.5	4.073	3.803	3.42	3.899	3.553	3.326	2.991	3.17	3.007
std avg		4.225	3.937	3.548	4.058	3.687	3.446	3.094	3.284	3.115
		0.12	0.096	0.091	0.095	0.088	0.088	0.099	0.087	0.096

Figure 5.5: Illustrating the compression ratio in bpp for the proposed models compared to the state-of-the-art lossless compression methods on TestSet2 (16-bits volumes). The first column is colour mapped by the pixel spacing value of each volume. The other cells are highlighted from the maximum compression 2.949 bpp (Blue) to the minimum compression of 4.52 bpp (Red).

TestSet2. To demonstrate the robustness and generalization ability of our proposed LSTM models, we evaluate our models’ performances on an out of the domain public dataset with a different modality (MRI) and represents a different part of a

patient’s body (head and neck). We also compare against a deep learning lossless compressor known as LSTM-Compress. The evaluation over TestSet2 is illustrated in Fig. 5.5. The first column illustrates the pixel spacing of TestSet2, which is the same for all volumes (0.5). Each other column represents the bits per pixel (bpp) of a lossless compression method, including PPMd, JPEG-LS, JPEG2000, HEVC, JP3D, LSTM-Compress, MedZip1, MedZip2, and MedZip3, respectively. The last row is the average (bpp) for each method through all volumes. Cells in other columns are highlighted from the maximum compression 2.949 bpp (Blue) to the minimum compression of 4.52 bpp (Red). Among the standard codecs, JPEG2000 yields the best bit-per-pixel results, followed by JP3D. Unexpectedly, HEVC, which can exploit inter-frame redundancy, gains only better compression than PPMd. Additionally, the PPMd algorithm has the worst compression compared to the other methods. Compared to the state-of-the-art techniques, the deep learning approach (LSTM-Compress) achieves better compression. Overall, the proposed RNN approaches gained the best compression bit-rate when compared to the standard codecs and the deep learning method (LSTM-Compress). Although our models were not trained on this dataset, they have still obtained the best bpp reduction.

In comparison to the deep learning method (LSTM-Compress), our proposed MedZip models demonstrate the best compression ratio and reduce compression time, showing that using 3D neighbourhood information works well to predict voxel values. MedZip models are notably faster than LSTM-Compress (2.26 hours vs 56.66 hours) to compress all of the volumes in Testset2. The main limitation of LSTM-Compress is due to the non-deterministic framework, which prevents parallel implementation and, thus, slower coding (for further details on this limitation, refer to subsection 2.4.5.2). In terms of compression ratio over Testset2, MedZip has a size reduction of up to 11% compared to LSTM-Compress. Since each volume takes around 30 hours to be compressed by LSTM-Compress, we compared the compression performance of LSTM-Compress on seven volumes selected randomly from TestSet1. The average compression performance of MedZip3 on those volumes is 4.50 bpp while LSTM-Compress is 5.13 bpp, with MedZip3 showing a saving of 12% over LSTM-Compress.

In Fig. 5.6, a summary of the compression performance over the two test sets is computed using the percentage change for each lossless method. Each row represents the performance of a single compression method over different datasets. Cells are coloured from the maximum compression percentage 100.00% (Blue) to the minimum performance of 136.56% (Red) – (Less value indicates better performance). Among the existing codecs, JP3D proposes the best compression performance on TestSet1.

However, over TestSet2, JPEG2000 gains the best reduction. An unexpected performance was given by the 3D codec HEVC, which produces better compression than PPMd on all datasets. Although the compression level of PPMd was set to Ultra setting, this algorithm performs the worst result on average on both datasets. By observing the results of Fig. 5.6, the overall compressing performance of our MedZip models illustrates the best reduction over the different datasets. MedZip3 outperforms MedZip2 with better reduction up to 5%. The advantage of MedZip1 is more noticeable, reaching a gain of almost 36% over the PPMd. To conclude, MedZip3 performs the state-of-the-art compression percentage of 100.00% on 16-bits volumetric medical images.

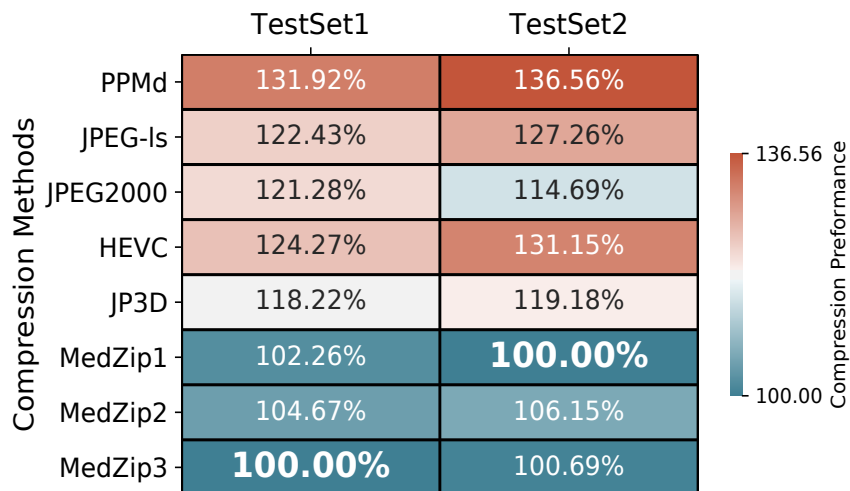


Figure 5.6: A summary overview of the compression performance over the two test sets for all the lossless methods. Cells are coloured from the best compression performance 100.00% (Blue) to the worst performance 136.56% (Red). (Less value indicates better performance).

5.4 Conclusion and Further Work

This chapter proposed a novel lossless compression approach using LSTM, specifically for compressing 3D medical images (16 bit-depths). A solution for the lossless compression problem was presented using the LSTM-based sequence prediction model and 3D pyramid-shaped sequences as the model’s input. MedZip empirically demonstrates a reduction in the compressed size (bpp) compared to the state-of-the-art lossless compression methods. Additionally, our pre-trained LSTM model generalized well to unseen modality (MRI) and achieved a higher compression ratio than the other

methods. Finally, although the current case study focuses on 3D medical images, we hypothesize that our approach can further be extended to other volumetric datasets such as video. We also believe that the proposed models would achieve more improvement by integrating them with attention-based mechanisms.

In the following chapter, we extensively analyse the impact of various strategies for extracting the model’s input sequences and training batches specifically for compressing (16-bit depth) volumetric medical scans losslessly. The intention is to determine the option with the best balance between compression time and compression ratio.

Receptive Field Sampling for Learning-based 3D Medical Image Compression



Contents

6.1	Introduction	154
6.1.1	Contributions	155
6.2	Related Work	155
6.2.1	Classical Prediction Based Methods	156
6.2.2	Learning Based Prediction Methods	157
6.3	Proposed Method	160
6.3.1	Problem Description	160
6.3.2	Causal Neighbouring Sequence	160
6.3.3	Model	162
6.3.3.1	Training Hyper Parameters	162
6.3.4	Benchmarks	163
6.3.4.1	Benchmark1: Optimal Input Sequence (shape and size)	163
6.3.4.2	Benchmark2: Optimising Decoder Performance . . .	164
6.3.4.3	Benchmark3: Encoding-Decoding Performance . . .	165
6.3.4.4	Benchmark4: Sampling Strategy	165
6.3.4.5	Benchmark5: Compression Improvement During Train- ing	165
6.3.4.6	Benchmark6: Comparing with State-of-The-Art Lossless Compression Methods	166
6.4	Results and Discussion	166
6.4.1	Dataset	166
6.4.2	Performance Metrics	168
6.4.3	Benchmark1: Optimal Input Sequence (shape and size)	168
6.4.4	Benchmark2: Optimising Decoder Performance	172
6.4.5	Benchmark3: Encoding-Decoding Performance	175
6.4.6	Benchmark4: Sampling Strategy	177
6.4.7	Benchmark5: Compression Improvement During Training . . .	178
6.4.8	Benchmark6: Comparing with State-of-The-Art Lossless Com- pression Methods	179
6.4.9	Residual Plots	181
6.5	Conclusion and Further Work	186

This work was originally accepted for Elsevier Machine Learning with Applications (MLWA) Journal, 2022, [45], by the thesis author alongside Dr J. Whittle, Dr J. Deng, Prof. B. Mora, and Prof. M. W. Jones.

6.1 Introduction

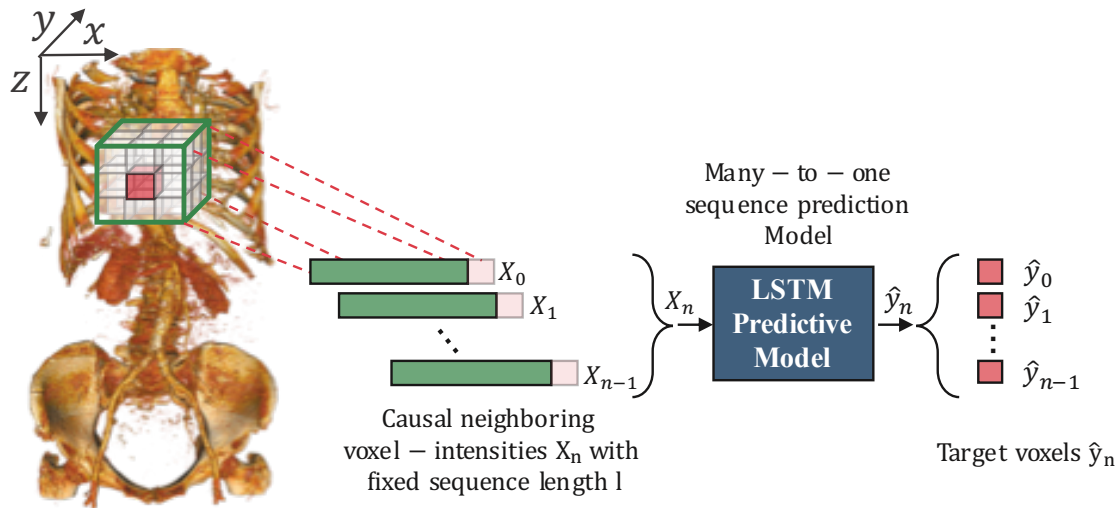


Figure 6.1: Illustration of the supervised learning LSTM model with an explicit overview of the method for extracting the causal neighbouring sequence from 3D medical images (16 bit-depth).

As discussed in the previous chapters 4 and 5, we proposed a novel learning-based codec as a sequence prediction model and improved its generalisability across different scanner settings and quality. This chapter broadly explores and analyses various input sequence options while empirically highlighting trade-offs and studying the impact on compression performance (i.e. compression time and ratio). Also, in this chapter, we seek to leverage parallelism within decoding to reduce compression time.

In the current research trends within the compression field, precisely the predictive compression scheme, 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 [43, 44]. This chapter examines further options for the causal neighbouring sequence and determines the compression

trade-offs between size and speed. 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.

6.1.1 Contributions

This chapter 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 are as follows:

1. We present recurrent neural network (LSTM) learning-based models for compressing 16-bits medical data and offer parallelisation of the decoder resulting in a speed-up of up to $37\times$ 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 chapter is organized as follows: Section 6.2 outlines the current state-of-the-art approaches for lossless compression and deep learning predictive models. Section 6.3 describes the proposed methodology, model’s architectures, training settings, and evaluation benchmarks. The experimental results are introduced and discussed in Section 6.4 with compression results of our proposed models compared to the state-of-the-art lossless compression methods across multiple benchmarks. The outcomes and conclusions are reported in Section 6.5.

6.2 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 [97–99]. 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 [133]. According to current compression research

trends, reduction methodologies can be classified into four main types: prediction-based [9, 10, 13, 43, 44, 134–141], transform-based [7, 142], quantisation-based [143, 144], and end-to-end compression frameworks [147]. This chapter 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.

6.2.1 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 [13]. 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) [134]. Minimum-Rate Predictor (MRP) has a wider causal neighbourhood and is an adaptive predictive-based method that applies 2D-block classification [135]. 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) [136] and (M-CALIC) [137] 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 [10].

A well-known 3D codec for video compression is High Efficiency Video Coding (HEVC) [9], 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 [159] 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 domains 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 the flexibility to represent non-linear data distributions.

6.2.2 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 [160–162]. This shifting toward the learning-based methods is consequent to its outstanding performance in many domains, exceptional ability to represent non-linearity, and GPU utilisation. Numerous learning-based approaches are proposed for lossy compression for applications including image-super resolution [176], dimensionality reduction (autoencoders) [148], generative compression [169], and end-to-end compression frameworks [147]. An autoregressive model is one of the state-of-the-art models in estimating data distribution and pixels likelihood. In PixelRNN [155], 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 being relatively slow due to the sequential implementation. However, the PixelCNN [155] provides a parallelised version, employing a smaller receptive field but not fully utilising the available context. PixelCNN++ [156] and Multiscale-PixelCNN [157] 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 [120]. 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) [122], Long Short-Term Memory (LSTM) [123], Gated Recurrent Units (GRU) [124], and Transformers [121, 131, 132].

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 [138]. Their method also includes a residual error block to further exploit the pixel’s inter-prediction and a novel context-based bit entropy coder outperforming 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 [139]. Another image codec that manipulates neighbouring pixels known as a channel-wise progressive prediction was presented in [141]. 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 [140]. 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 a predictive scheme but for sequential data are DeepZip [145], LSTM-Compress [146], and Dzip [3]. 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 are 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 [146] (see section 6.4 for more details).

In the previous chapters 4 and 5, we proposed learning-based predictive methods, which support 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 3D contextual neighbouring voxels to predict a single target voxel. This chapter 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 chapter 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.

6.3 Proposed Method

6.3.1 Problem Description

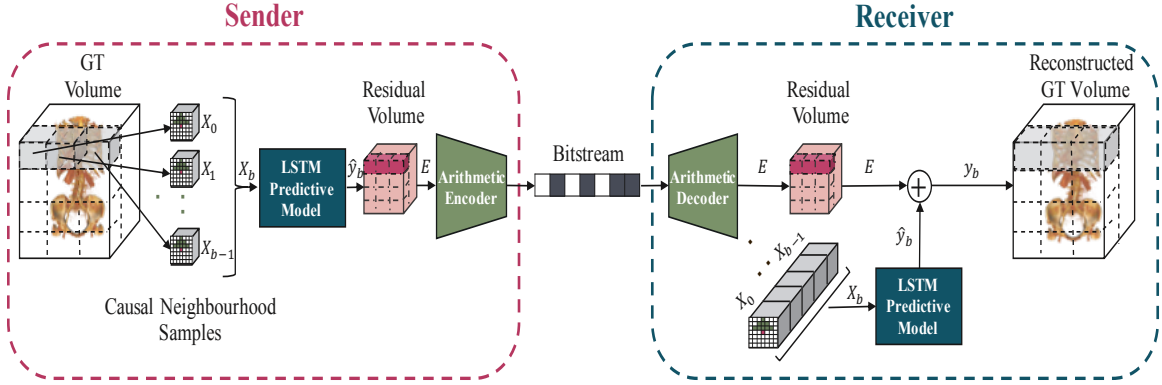


Figure 6.2: Overview of the optimized version of our lossless compression framework using LSTM. This version leverages parallelism by employing a novel input sequence, which adds more flexibility and allows the decoder to process several batches of sequences in parallel.

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 $X_i = \{x_0, x_1, x_2, \dots, x_{l-1}\}$ with a fixed sequences' length l as illustrated in Figure 6.1. 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 minimize 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 a 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. 6.2 for more details).

6.3.2 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 sequence 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 sequences extracted from 3D high-resolution medical volumes. Two different block shapes were introduced: the cube and the pyramid shapes, as illustrated in Figure 6.3. For each shape, several sizes (various distances to the target voxel) were applied. Moreover, aggregation of surrounding voxels form different dimensions were applied, including 1D, 2D, and 3D coverage.

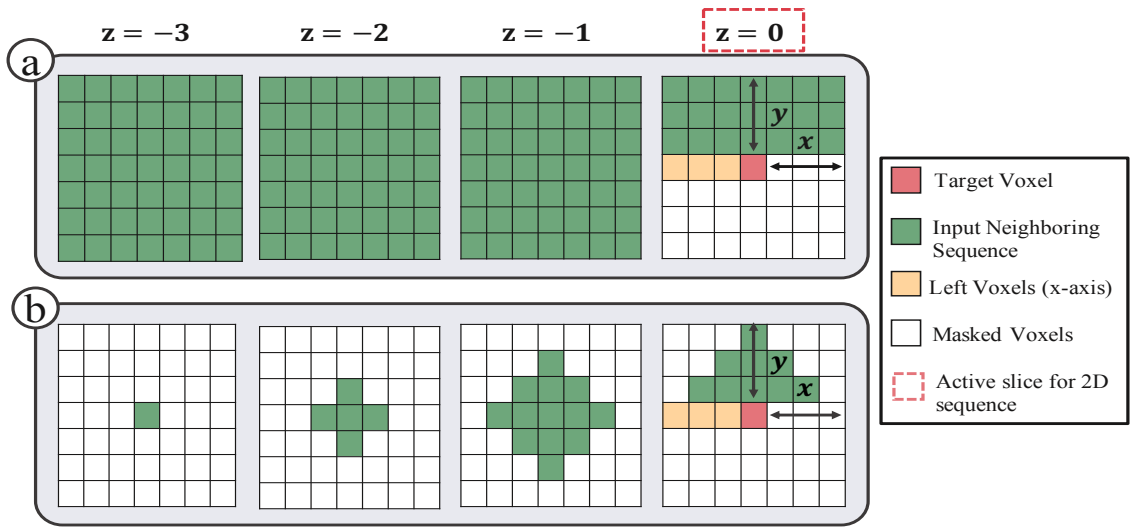


Figure 6.3: A demonstration of the proposed causal neighbouring sequences that can have different shapes, dimensions, block sizes, and sequence lengths. The two main shapes are (a) Cube and (b) Pyramid. The red voxel refers to the target voxel to be predicted while the green voxels form the model’s input sequence and the white voxels are masked (excluded). $z = 0$ represent 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 6.3.4.1, but omitted from the input sequence of experiment 6.3.4.2.

6.3.3 Model

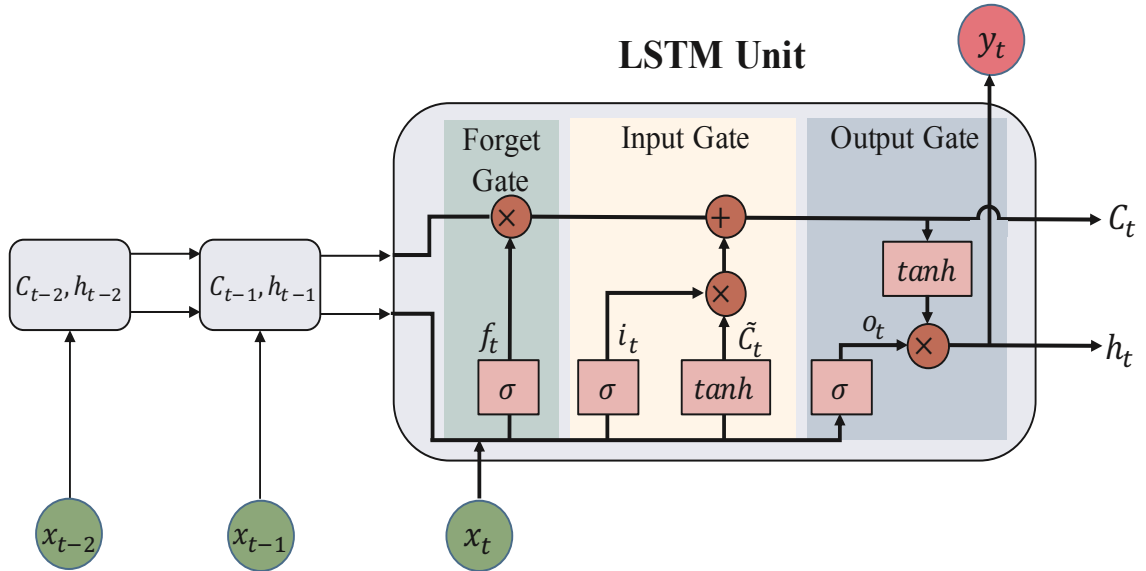


Figure 6.4: A many-to-one standard LSTM model with a detailed overview of the LSTM unit. Each LSTM cell contains three main gates (input i_t , forget f_t , and output gate y_t), and a cell state C_t .

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. An illustration of a many-to-one LSTM predictive model with a detailed overview of the cells' gates is provided in Figure 6.4.

6.3.3.1 Training Hyper Parameters

Layer	Number of Neurons	Activation Function Used
LSTM	128	Sigmoid and Tanh
Output	1	Linear

Table 6.1: The NN architecture used as the standerd specification for all the proposed models.

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 (as shown in Table 6.1). The storage size required for the model's weights only is 276 KiloBytes (KB), while the complete model's size

(weights and training hyperparameters) is 810 Kilobytes (KB). **Optimizer:** We use Adam optimizer [177], 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:** Details on the loss function L_{joint} used for training all the models are provided in section 3.2.

6.3.4 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 a high compression ratio and fast encoding-decoding performance for 3D medical image reduction. All models used in this chapter 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 using 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**.

6.3.4.1 Benchmark1: 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. In comparison, the 2D case has four options: two with a cube shape and another two for the pyramid shape, each with (13×13) and (11×11) 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 Figure 6.3, 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). $z = 0$ represents 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 inputting them to the LSTM model.

6.3.4.2 Benchmark2: Optimising Decoder Performance

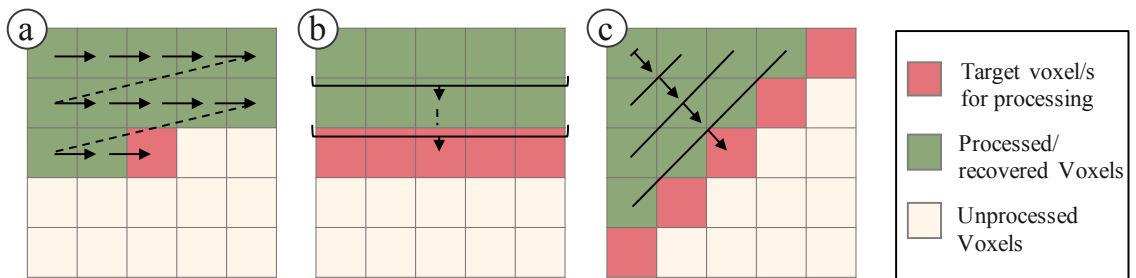


Figure 6.5: The order in which the decoder will decode voxels based on the defined causal neighbourhood specifications. (a) The order of decoding voxels (sequentially) restricted by neighbourhood dependencies applied in 6.3.4.1. (b) The order of decoding voxels (in parallel) by decoding an entire batch applied in 6.3.4.2. (c) a possible parallel approach to decoding voxels, which does not allow full GPU occupancy.

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 [3, 145] 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 [43, 44] 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. 6.5 (a). A benefit of our 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 subsection 6.3.4.1. In these sequences, the left x-axis voxels

(i.e. yellow voxels in Fig. 6.3) are removed from the input sequence. This allows the decoder to process batches of sequences in parallel, as illustrated in Fig. 6.5 (b). Removing the left voxels forms a simple but effective strategy to leverage parallelism compared to decoding voxels diagonally Fig. 6.5 (c). Parallel implementation using diagonal voxels does not lead to optimal GPU occupancy compared to our approach and complicates implementation.

6.3.4.3 Benchmark3: 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.

6.3.4.4 Benchmark4: Sampling Strategy

Approximately one billion voxels are available in the dataset, making training on all 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, $(13 \times 13, 9 \times 9, 5 \times 5, 1)$, $N = 175$) 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 toward 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.

6.3.4.5 Benchmark5: 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 (i.e. 3D pyramid shape with left voxels) 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 $(13 \times 13, 9 \times 9, 5 \times 5, 3 \times 3, 1)$.

6.3.4.6 Benchmark6: Comparing with State-of-The-Art Lossless Compression Methods

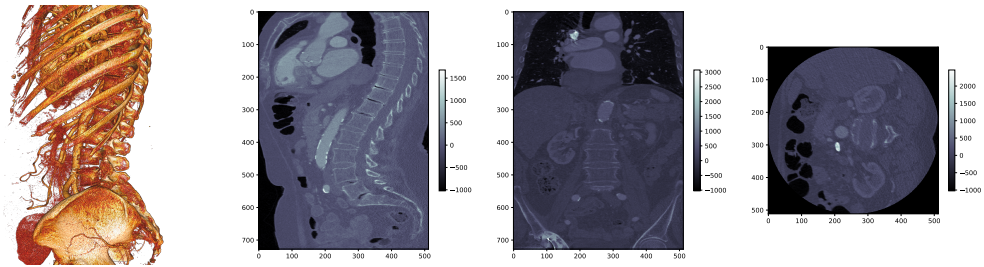
In this benchmark, the trained models with the best compression results from benchmarks 6.3.4.1 and 6.3.4.2 are compared to existing state-of-the-art approaches for lossless compression of volumetric medical data. Compression ratios for all compared approaches are reported using bits per pixel over the two available datasets.

6.4 Results and Discussion

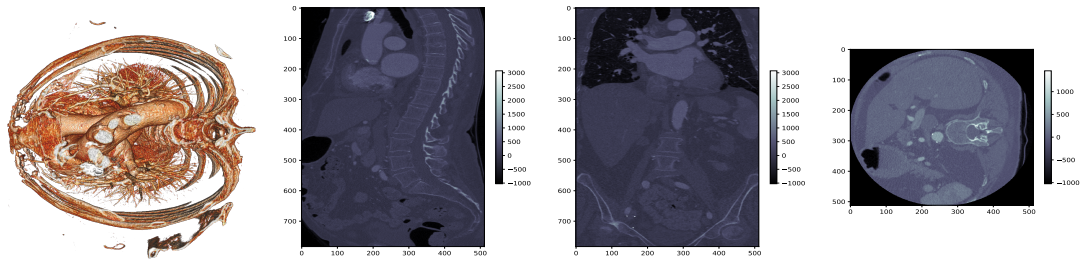
6.4.1 Dataset

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**. All scans have $[.488, .488]$ pixel spacing and $.625mm$ slice thickness (for further details on the dataset specifications, refer to section 3.1.1). An illustration of orthogonal slice views and 3D volume rendering of two sample volumes from **Dataset1** are presented in figure 6.6 (a) and (b).

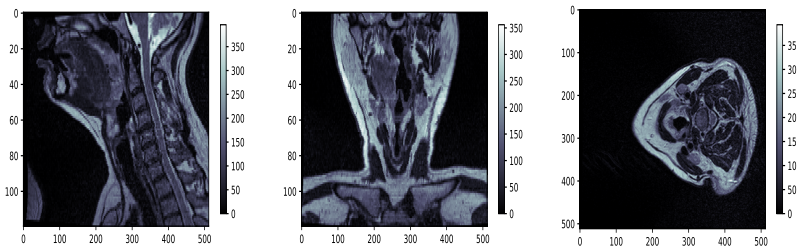
Another dataset, which was involved only for conducting experimental evaluation, is a public dataset known as **Dataset2**. **Dataset2** contains DICOM files that form a total of 12 MRI volumes of patients' head and neck scans with $[.5, .5]$ pixel spacing, and $2mm$ slice thickness [16–18] (see section 3.1.3 for further details on the dataset specifications). An illustration of orthogonal slice views of two sample volumes from **Dataset2** is shown in Fig. 6.6 (c) and (d).



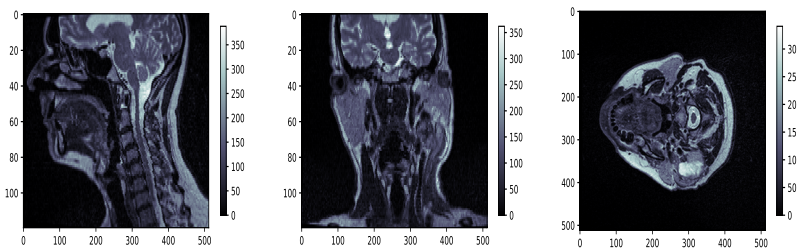
(a) Volume1 from Dataset1



(b) Volume6 from Dataset1



(c) Volume1 from Dataset2



(d) Volume12 from Dataset2

Figure 6.6: A visualisation of some (16-bits) sample volumes from the two datasets used in this chapter, namely, **Dataset1** and **Dataset2** is shown. Figure 6.6 (a) and (b) illustrates 3D volume visualization of patient's entire Torso with three orthogonal slice views (axial, sagittal and coronal) of two sample volumes from **Dataset1**. While Fig. 6.6 (c) and (d) presents three orthogonal slice views (axial, sagittal and coronal) of sample MRI volumes from **Dataset2** illustrating the patient's head and neck.

6.4.2 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 bpp (Eq. 3.3.1) has been chosen as the evaluation metric of the compression ratio obtained by all our LSTM models. In addition to evaluating compression time (in seconds), measured per model for both the encoding and the decoding operations (3.3.2).

6.4.3 Benchmark1: Optimal Input Sequence (shape and size)

This experiment aims to determine the best input sequence (shape and size) that would lead to optimal compression for 3D medical images. Figure 6.7 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 2D slices gain better compression performance for both cube and pyramid shapes. The 2D pyramid 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 with the smallest block size with just 22 voxels. This input vector allows its LSTM model to gain comparable compression results 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 from 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.

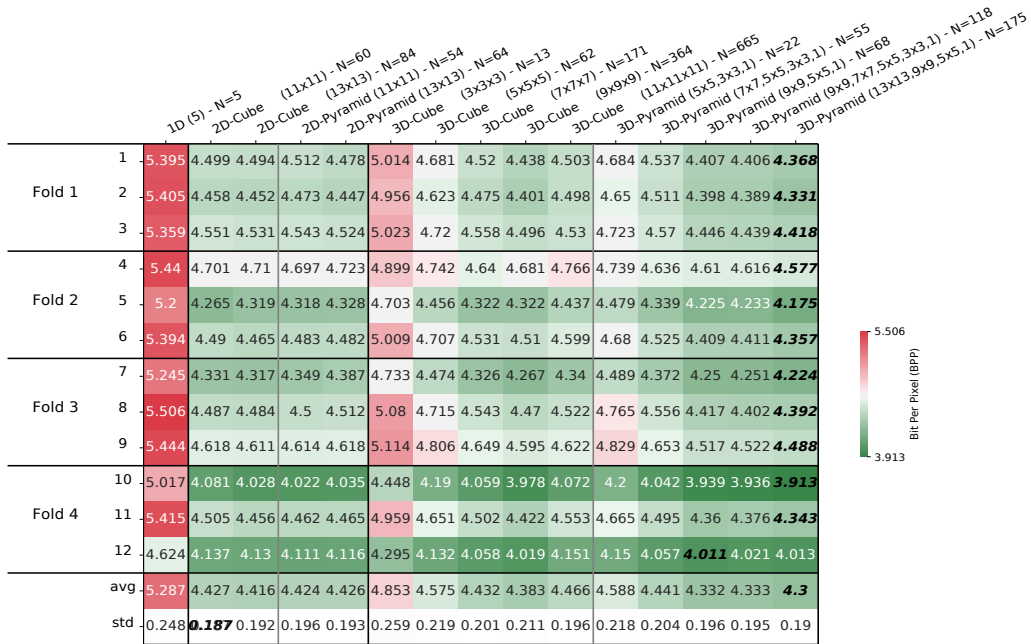


Figure 6.7: Benchmark 1 result that illustrates the compression ratio in Bits-per-pixel (bpp) for compressing **Dataset1** using models trained on different neighbouring sequences applied in Benchmark 1 section 6.4.3. The top labels specify the input sequences’ specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 3.913 bpp (Green) to the lowest compression 5.506 bpp (Red). (Another version of this figure with a different reordering is presented in Appendix B - Fig. B.2)

		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 (13x13x13) - N=222	3D-Pyramid (15x15x15) - N=55	3D-Pyramid (17x17x17) - N=68	3D-Pyramid (19x19x19) - N=175	
<hr/>																
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		3.108	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
std		0.098	0.093	0.094	0.096	0.094	0.099	0.092	0.088	0.088	0.092	0.097	0.09	0.089	0.089	0.093
<hr/>																
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
std		0.097	0.094	0.094	0.094	0.096	0.098	0.089	0.092	0.094	0.098	0.095	0.09	0.087	0.092	0.09
<hr/>																
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
std		0.098	0.093	0.096	0.096	0.092	0.097	0.089	0.087	0.092	0.096	0.094	0.09	0.091	0.089	0.089
<hr/>																
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
std		0.099	0.092	0.094	0.095	0.091	0.1	0.088	0.085	0.087	0.095	0.094	0.089	0.088	0.086	0.089
<hr/>																
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
	std	0.101	0.092	0.094	0.097	0.097	0.096	0.088	0.086	0.105	0.093	0.092	0.09	0.09	0.092	0.088

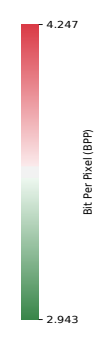


Figure 6.8: Compression performance in (bpp) for evaluating the trained models with different neighbouring sequences from Benchmark 1 section 6.4.3 validated over **Dataset2**. Models pre-trained on distinct neighbouring sequences from **Dataset1** are evaluated over this unseen set and grouped into 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).

Figure 6.8 presents a validation of all trained models over **Dataset2**. Models pre-trained on distinct input configurations from **Dataset1** are evaluated on this set 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 average bpp achievements overall folds' means. Although our proposed models were not trained on this dataset, they still generalise well to all volumes belonging to this data. Cells are coloured from maximum compression 2.9432 bpp (Green) to the lowest compression 4.247 bpp (Red). 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 pyramid 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 $[\cdot488, \cdot488]$ and evaluated on $[\cdot5, \cdot5]$). While in the case of 3D quality, there is a variation in the slice thickness between what the models learned from (i.e. $\cdot625mm$) and validated on (i.e. $2mm$ 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 and yet better compression results achieving a mean of 3.123 bpp by the best compressor. Overall, the proposed models with 2D pyramid (11×11), 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 3.081bpp, 3.079bpp, and 3.11bpp, respectively. From observing the evaluation results in 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, as will be shown in Fig 6.15.

6.4.4 Benchmark2: Optimising Decoder Performance

		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-Pyramid (5x5,3x3,1) - N=659	3D-Pyramid (7x7,5x5,3x3,1) - N=19	3D-Pyramid (9x9,7x7,5x5,3x3,1) - N=51	3D-Pyramid (13x13,9x9,5x5,1) - N=112	3D-Pyramid (13x13,9x9,5x5,1) - N=169	
Fold 1	1	4.779	4.806	4.748	4.766	5.302	4.968	4.769	4.696	4.749	4.859	4.776	4.64	4.652	4.634
	2	4.705	4.757	4.699	4.705	5.203	4.871	4.69	4.636	4.691	4.777	4.708	4.582	4.612	4.573
	3	4.857	4.866	4.819	4.845	5.331	5.041	4.842	4.78	4.802	4.925	4.826	4.682	4.698	4.68
Fold 2	4	4.861	4.88	4.871	4.894	5.111	4.902	4.769	4.803	4.752	4.873	4.799	4.793	4.801	4.768
	5	4.545	4.541	4.548	4.546	4.966	4.684	4.51	4.496	4.421	4.621	4.527	4.437	4.453	4.404
	6	4.725	4.729	4.73	4.715	5.285	4.969	4.765	4.728	4.632	4.858	4.768	4.654	4.676	4.597
Fold 3	7	4.513	4.485	4.519	4.509	4.973	4.703	4.526	4.448	4.443	4.635	4.534	4.424	4.487	4.406
	8	4.721	4.686	4.723	4.715	5.34	4.979	4.778	4.684	4.621	4.847	4.777	4.616	4.691	4.594
	9	4.893	4.837	4.878	4.865	5.412	5.137	4.931	4.832	4.77	5.012	4.894	4.758	4.785	4.731
Fold 4	10	4.228	4.194	4.215	4.212	4.708	4.384	4.19	4.131	4.114	4.324	4.178	4.109	4.131	4.051
	11	4.714	4.676	4.706	4.698	5.218	4.898	4.712	4.656	4.62	4.792	4.684	4.581	4.625	4.541
	12	4.269	4.254	4.274	4.254	4.52	4.356	4.224	4.192	4.174	4.326	4.21	4.181	4.18	4.148
	avg	4.651	4.643	4.644	4.644	5.114	4.824	4.642	4.59	4.566	4.738	4.64	4.538	4.566	4.511
	std	0.22	0.23	0.216	0.224	0.275	0.247	0.235	0.23	0.23	0.221	0.235	0.214	0.217	0.222

Figure 6.9: Benchmark 2 result that illustrates the compression ratio in (bpp) for compressing **Dataset1** using models trained on **reduced** neighbourhood sequences applied in section 6.4.4. 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). (An additional version of this figure with a different reordering is presented in Appendix B - Fig. B.3)

This experiment introduces new sequences that drop the left neighbouring voxels in order to achieve parallel decoding. Figure 6.9 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 models’ performance trained on a specific sequence overall volumes. The last row presents the average bpp per model across all volumes. Cells are coloured from maximum compression 4.051 bpp (Green) to least 5.412 bpp (Red). The best compression result is in bold. The 1D input sequence is not included since it is not applicable. The models trained on 2D neighbourhood sequences all produce similar results, with the pyramid-shaped sequences having lower costs since fewer voxels are used. The models trained on cubic shaped input demonstrate better compression as block size increases. The model trained with (3^3) has the least compression at 5.114 bpp, while the (11^3) block size demonstrates 4.566 bpp. The models trained on sequences based on a 3D pyramid achieve the best reductions

regardless of block size. The sequences with $N = 61$ and $N = 169$ voxels produce the best bit reduction among all the models on **Dataset1**. Overall, this novel input strategy’s performance has a drop of ≈ 0.2 bpp in compression ratio compared to the non-parallel version (Figure 6.7), 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 6.4.5). A summary overview of the average storage impact of the method bpp over all 12 volumes for the two experiments is presented in Fig. 6.10.

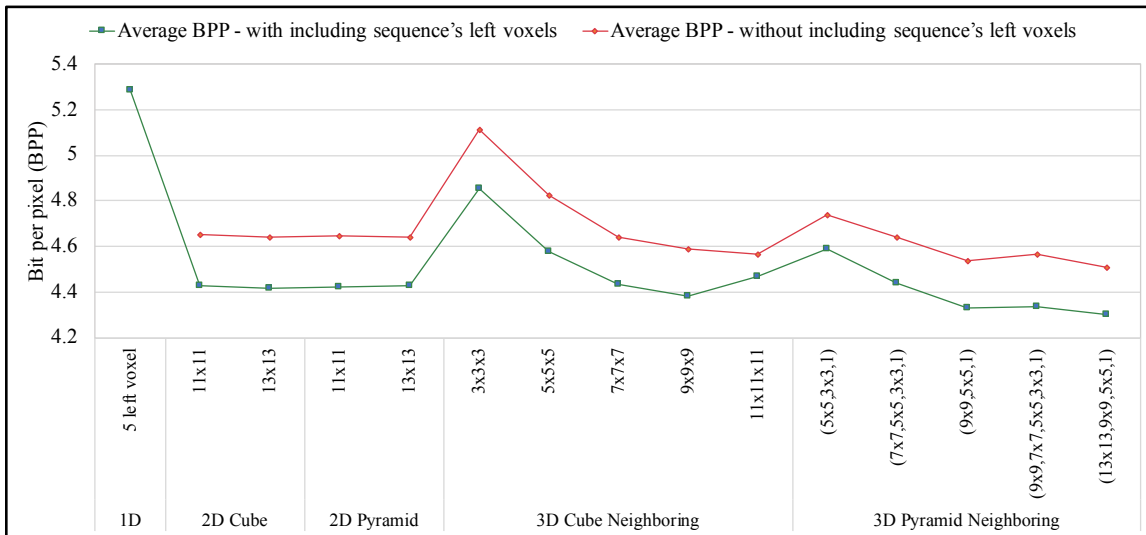


Figure 6.10: A summary overview of the average (bpp) over Dataset1’s volumes using all models trained with left voxels (in Green) and without (in Red) applied in Benchmark 1, and Benchmark 2, respectively. Overall, models pre-trained on reduced inputs result in a performance drop of ≈ 0.2 bpp, but with a significant positive impact on the compression times (see section 6.4.5 for further details). Additional version of this figure with a different reordering is presented in Appendix B – Fig B.4.

		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-Pyramid (11x11x11) - N=659	3D-Pyramid (5x5x3x3x1) - N=19	3D-Pyramid (7x7x5x3x3x1) - N=51	3D-Pyramid (9x9x5x5x1) - N=61	3D-Pyramid (13x13x9x9x5x5x1) - 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
std		0.103	0.103	0.103	0.104	0.11	0.102	0.099	0.102	0.104	0.097	0.102	0.099	0.097	0.1
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
std		0.103	0.106	0.101	0.1	0.107	0.1	0.099	0.099	0.101	0.09	0.106	0.099	0.101	0.098
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
std		0.102	0.101	0.099	0.101	0.109	0.103	0.102	0.104	0.099	0.098	0.101	0.097	0.1	0.095
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
std		0.098	0.101	0.097	0.098	0.107	0.1	0.099	0.1	0.101	0.094	0.097	0.092	0.096	0.096
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
	std	0.099	0.101	0.098	0.098	0.106	0.098	0.098	0.101	0.098	0.098	0.102	0.099	0.097	0.098

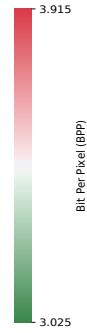


Figure 6.11: Compression performance in (bpp) for evaluating models trained on the **reduced** neighbourhood sequences applied in section 6.4.4 on **Dataset2**. The top labels specify the input sequences' specifications, including dimensions, shape, block size, and sequence length, respectively. Cells are coloured from maximum compression 3.025 bpp (Green) to the lowest compression 3.915 bpp (Red).

In figure 6.11, 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**). Cells are coloured from maximum compression 3.025 bpp (Green) to the lowest compression 3.915 bpp (Red). 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** .625mm) and the testing data (i.e. **Dataset2** 2mm). However, it is interestingly noticeable that many of the 3D input sequences still achieve a significant compression performance of 3.176 bpp 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, respectively. 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. As was indicated in the previous Fig. 6.9, models trained with the reduced version of neighbouring sequences are expected to have a drop in the compression performance (i.e. $\approx .2$ bpp over **Dataset1**) since the input lacks a number of the spatial voxels' values. However, in **Dataset2**, the performance drop affected by the novel version of input formats is only $\approx .11$ bpp, which achieves encoding-decoding speed-up and affords a significant gain to the overall compression rate.

6.4.5 Benchmark3: Encoding-Decoding Performance

Table 6.2, presents the time in seconds (s) required to compress and decompress a single slice belonging to **Dataset1** with all the different neighbourhood models. The inputs 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 6.2, one concludes that the 3D pyramids with $(13 \times 13, 9 \times 9, 5 \times 5, 1)$, $N = 169$, and $(9 \times 9, 5 \times 5, 1)$, $N = 61$ present the best balance of compression time and ratio (5.95s, 4.511 bpp), and (3.6s,

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 (11x11)	60	4.427	3.19	1127.65	54	4.651	3.12	3.21
2D Block (13x13)	84	4.416	3.73	1408.7	77	4.643	3.71	3.66
2D Diamond (11x11)	54	4.424	2.96	1066.61	48	4.644	2.89	3.07
2D Diamond (13x13)	64	4.426	3.17	1158.08	58	4.644	3.05	3.11
3D Cube (3x3x3)	13	4.853	2.33	627.91	11	5.114	2.09	2.16
3D Cube (5x5x5)	62	4.575	3.25	1167.93	59	4.824	3.22	3.56
3D Cube (7x7x7)	171	4.432	5.76	2445.81	167	4.642	5.72	5.86
3D Cube (9x9x9)	364	4.383	10.46	4776.11	359	4.59	10.24	10.44
3D Cube (11x11x11)	665	4.466	17.6	8384.63	659	4.566	17.61	17.67
3D Pyramid (5x5,3x3,1)	22	4.588	2.48	718.38	19	4.738	2.41	2.47
3D Pyramid (7x7,5x5,3x3,1)	55	4.441	3.39	1049.63	51	4.64	3.29	3.34
3D Pyramid (9x9,5x5,1)	68	4.332	3.5	1200.53	61	4.538	3.36	3.5
3D Pyramid (9x9,7x7,5x5,3x3,1)	118	4.333	4.9	1815.33	112	4.566	4.61	4.69
3D Pyramid (13x13,9x9,5x5,1)	175	4.3	6.14	2512.65	169	4.511	5.94	5.85
LSTM-Compress	–	–	–	–	–	4.891	133.22	132.32
Average	–	4.512	4.986	2004.409	–	4.665	5.09	5.185

Table 6.2: 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 causal neighbouring sequences (with and without the left voxels) from Benchmark1 and Benchmark2. The best compression result (bpp) on average, is in bold.

4.538 bpp), respectively.

Comparison to existing method: We also evaluated against a comparative existing and available deep learning method (LSTM-Compress [146]) 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.

6.4.6 Benchmark4: Sampling Strategy

	Slice-based Sampling	Random Sampling	Gaussain Sampling
1	4.3681	4.3548	4.4046
Fold 1 2	4.3311	4.3125	4.3779
3	4.418	4.4076	4.4374
4	4.5767	4.5105	4.5707
Fold 2 5	4.1754	4.1432	4.1928
6	4.3568	4.3269	4.3745
7	4.2237	4.2057	4.2668
Fold 3 8	4.3917	4.4196	4.4701
9	4.4875	4.5048	4.5542
10	3.9131	3.935	3.9039
Fold 4 11	4.3426	4.3546	4.3245
12	4.0125	4.0239	3.9939
avg	4.2998	4.2916	4.3226
std	0.19	0.181	0.206

Figure 6.12: 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).

Figure 6.12 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.

6.4.7 Benchmark5: Compression Improvement During Training

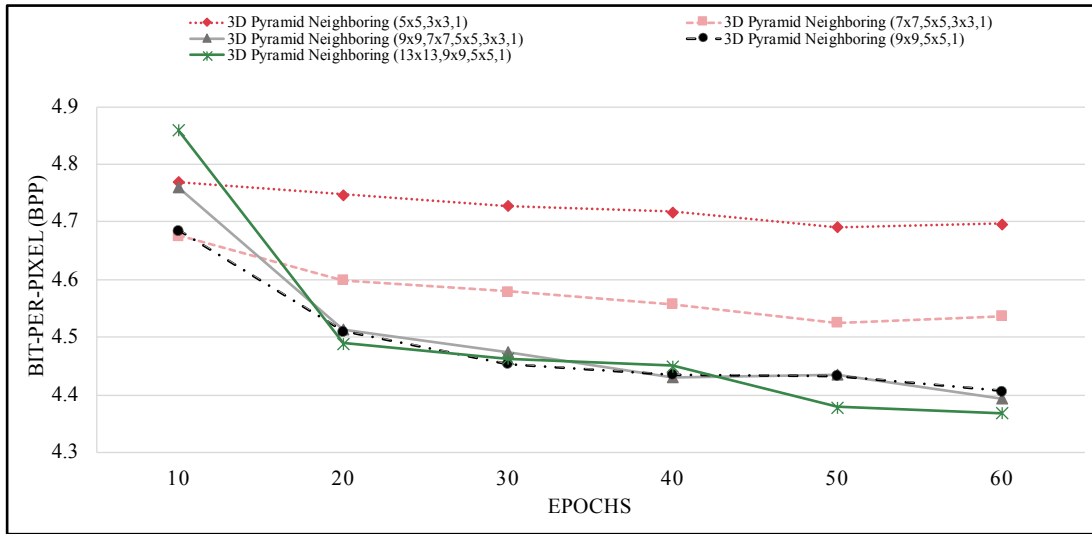


Figure 6.13: Benchmark5 result, which illustrates (bpp) variations during models’ training steps. To clarify, this plot does not demonstrate the model’s training loss function, but it evaluates trained models’ compression qualities after different epochs. Models with pyramid input vectors (including left voxels from Benchmark1) are evaluated after 10, 20, 30, 40, 50 and 60 epochs. (A different version of this figure that compares the loss function plot across these models is provided in figure B.5).

Figure 6.13 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 the model’s training loss function, but it evaluates trained models’ compression bpp after different epochs. Among these cases, the pyramid with $(13 \times 13, 9 \times 9, 5 \times 5, 1)$ block size obtains the best reduction followed by $(9 \times 9, 7 \times 7, 5 \times 5, 1)$, and $(9 \times 9, 5 \times 5, 1)$

6.4.8 Benchmark6: Comparing with State-of-The-Art Lossless Compression Methods

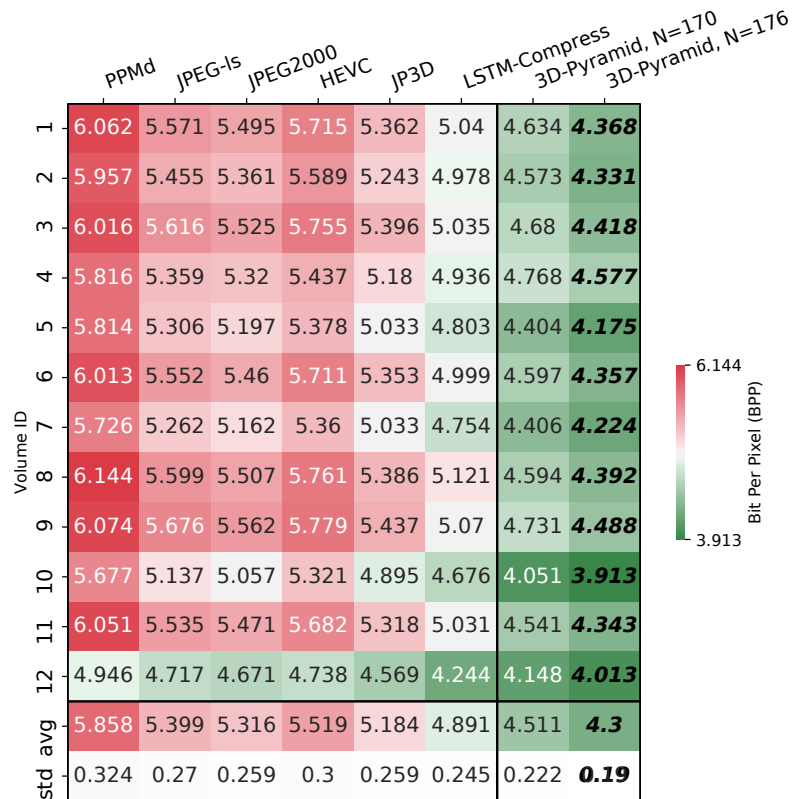


Figure 6.14: Benchmark 6 result that illustrates the compression ratio in bpp for two of the proposed models compared the state-of-the-art lossless compression methods over **Dataset1** (16-bits volumes). Cells are highlighted from maximum compression 3.913 bpp (Green) to minimum compression 6.144 bpp (red). Our two proposed models have 3D pyramid shapes with $(13 \times 13, 9 \times 9, 5 \times 5, 1)$ while $N = 170$ forms sequence without including left voxels, and $N = 176$ is sequence including left voxels

The experiment in Fig. 6.14 and Fig. 6.15 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 [175], JPEG-LS [173], JPEG2000 [174], HEVC [159, 172], JP3D [174], and the deep learning method LSTM-Compress [146].

By observing the results in figure 6.14, one noticed that among the standard codecs, 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.

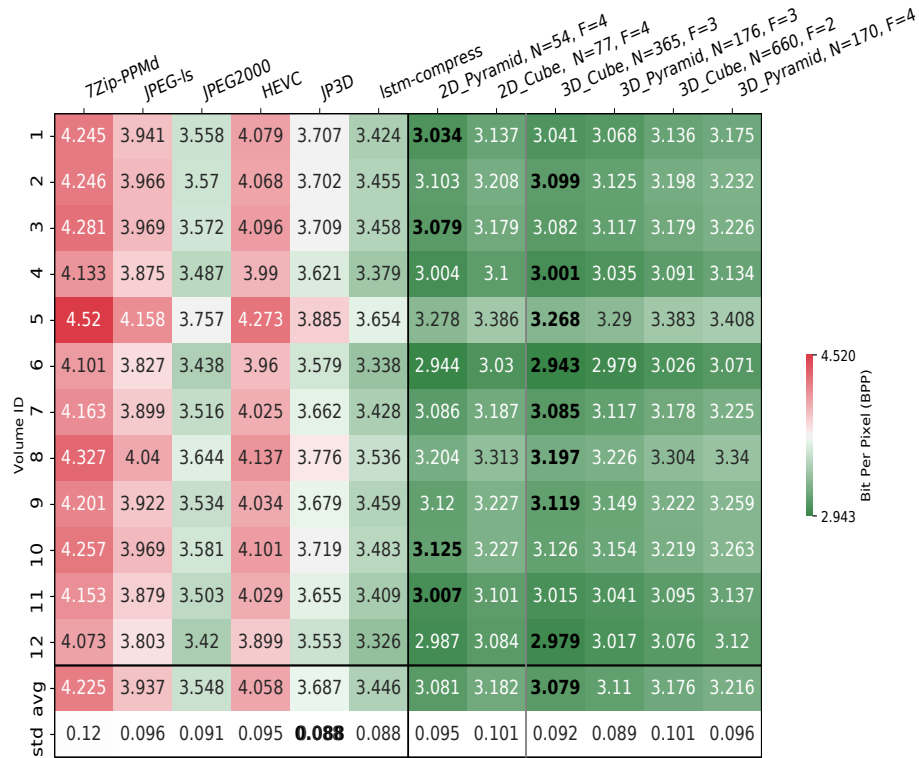


Figure 6.15: Illustrating the compression ratio in bpp for evaluating some of our pre-trained models compared to the state-of-the-art lossless compression methods over **Dataset2** (16-bits volumes). Cells are highlighted from maximum compression 2.943 bpp (Green) to minimum compression 4.52 bpp (red). 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").

In figure 6.15, 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. When comparing our proposed models' compression results to its competing codec LSTM-Compress, noticeable save in spaces around 10% and 7% are gained by our two state-of-the-art models trained on input sequence with 3D pyramid-shapes. 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.

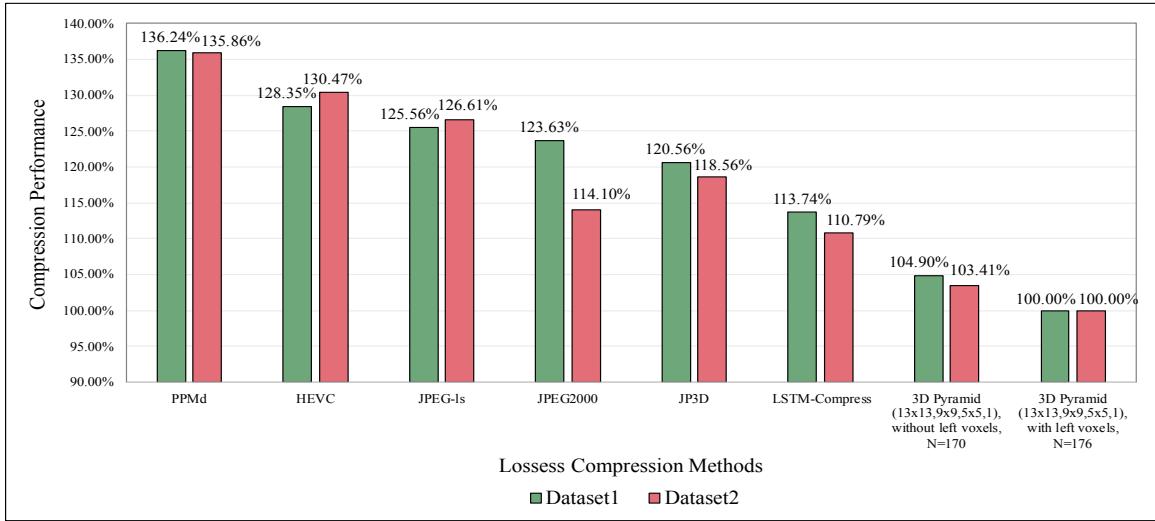


Figure 6.16: A summary overview of the compression performance over different 3D medical datasets (16-bits) (**Dataset1** and **Dataset2**) using the proposed input sequences of LSTM predictor models compared to the state-of-the-art lossless compression methods (Less value indicates better performance).

Overall, our one-step-ahead prediction models achieve state-of-the-art compression for all datasets with 17%, and 13% save in spaces over **Dataset1**, and 12%, and 9% saving in storage over **Dataset2** compared to the best performers among classical methods JP3D, and JPEG2000, respectively, as illustrated in Figure 6.16.

6.4.9 Residual Plots

In this subsection, we illustrate a number of residual comparisons plots from three random volumes belonging to **Dataset1**. 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 a similar technique) in order to provide lossless compression of the complete slice/data set. All figures contain comparisons among the different causal 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 also included under each sub-figure.

All residual slices are colour mapped with a 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. 6.17, 6.18, and 6.19 illustrates plots of middle residual slices selected randomly from volumes one, seven, and ten, respectively.

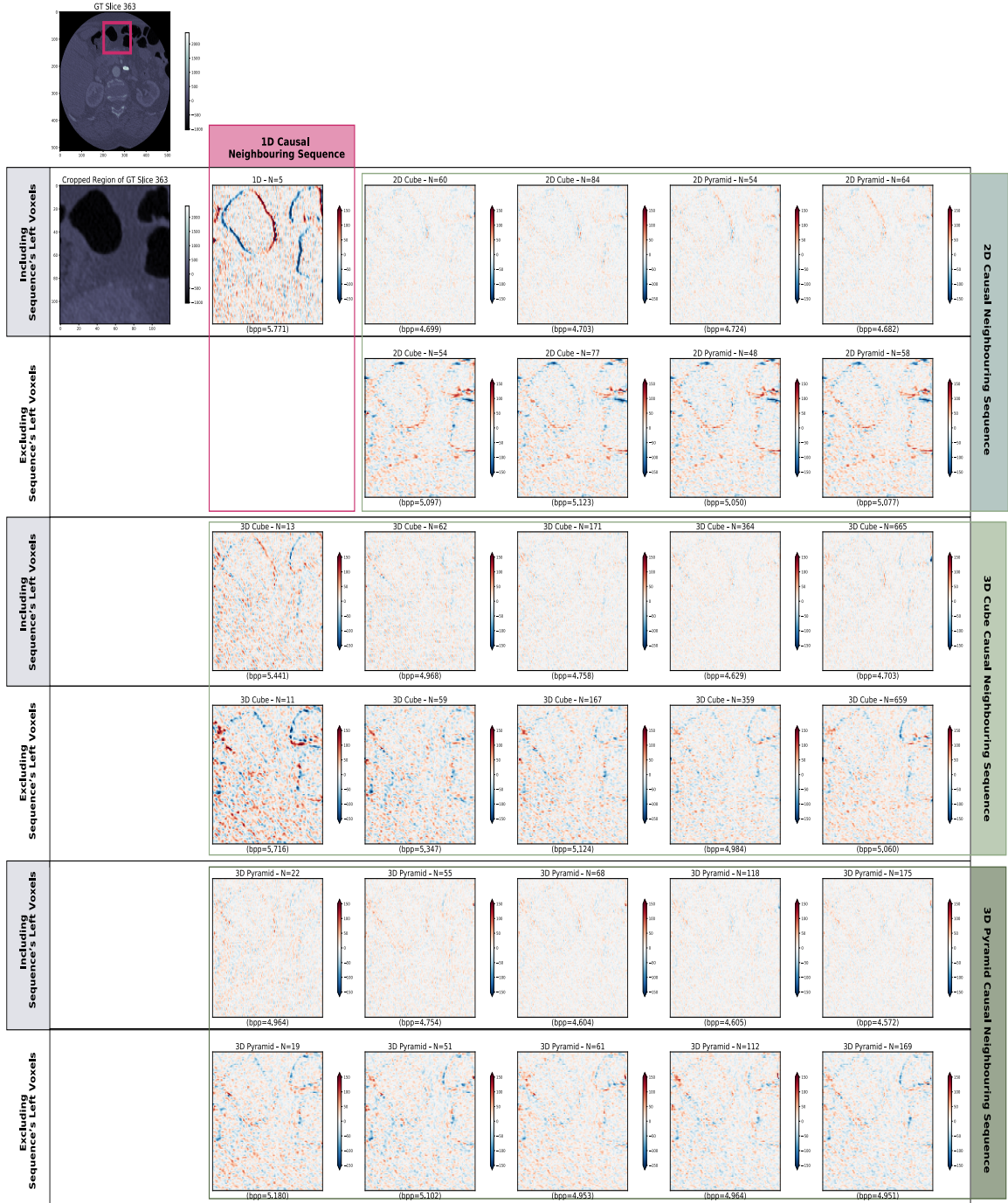


Figure 6.17: Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 1 while focusing on a specific region within the slice to highlight the impact. (Refer to Appendix B – Fig. B.6 to see the visualization of the whole residual slices).



Figure 6.18: Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 7 while focusing on a specific region within the slice to highlight the impact. (Refer to Appendix B – Fig. B.7 to see the visualization of the whole residual slices).



Figure 6.19: Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 10 while focusing on a specific region within the slice to highlight the impact. (Refer to Appendix B – Fig. B.8 to see the visualization of the whole residual slices).

6.5 Conclusion and Further Work

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, 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.

In the following chapter, we proposed novel approaches for extracting training batches from 3D medical scans (16-bit depth) while analysing the impact on the model's training stability and evaluation performance.

Importance Sampling Based on Data-driven Information



Contents

7.1	Introduction	189
7.2	Related Work	190
7.2.1	Contributions in Enhancing SGD Optimiser & Sampling Selection	191
7.2.2	Contributions in Sampling Quality & Scoring Metrics	192
7.3	Methodology	194
7.3.1	Problem Description	194
7.3.2	Alternative Sampling Strategies	195
7.3.2.1	Random Sampling	195
7.3.2.2	Gaussian Sampling	195
7.3.2.3	Slice-based Sampling	197
7.3.3	Proposed Sampling Method	197
7.3.3.1	Background	197
7.3.3.2	Gradient-based Sampling	199
7.4	Experimental Results and Discussion	199
7.4.1	Performance Metrics	199
7.4.2	Dataset Details	199
7.4.3	Model Details	203
7.4.4	Gradient-based Sampling	203
7.4.4.1	Gradient-based Sampling Applied to Dataset1	203
7.4.4.2	Gradient-based Sampling Applied to Dataset2	212
7.4.5	Comparing with State-of-The-Art Lossless Compression Methods	215
7.5	Conclusion and Further Work	217

7.1 Introduction

The task of the prediction-based model seeks to learn a mapping function from the given input to target output over a particular set of samples distributions. In such a model, it is desirable to include significantly more informative sequences within the training set, especially in domain tasks that contain up to billions of available sample options (e.g. volumetric medical scans). With such data distributions, and regardless of the specified deep learning task, the challenge that arises is the computation cost and feasibility of efficiently training on all available samples within the domain. Such a task is not restricted to medical datasets only; it forms one of the most challenging tasks in machine learning in many data application domains. As the number of samples grows, training steps scales linearly, with a higher chance to face training instability problems.

The increasingly massive scale of dataset generation resides the challenge of the deep learning training tasks. Based on the model’s size, the training time dramatically raises as the number of training samples increases. Not to mention the data variance and overfitting problems, which are affected by the quality and size of the samples [178]. The selection of training samples plays an essential role in the model’s training stability and overall performance. Choosing a representative subset of a dataset would speed up the training process while still retaining favourable learning achievements. The impact of sampling informative data would further extend to reduce the training time while improving stability.

We seek to determine whether it is possible to select a set of training sequences that is more representative than other sampling schemes (e.g. uniform sampling). We hypothesise that by training on more informative sequences that cover various data distributions, our many-to-one prediction model will produce better predictions with more compressible residuals than a similar model trained on a randomly sampled training set. We proposed novel data-driven sampling schemes using weighted gradient scores for training our LSTM prediction-based model. The developed selection procedure directly pre-chooses a fixed number of training sample points based on their gradient magnitudes and the specified thresholds. The performance of the proposed scheme was evaluated compared to models trained on various data-driven sampling schemes such as uniform, Gaussian, and sliced-based sampling to study its effectiveness and measure its quality.

Our proposed sampling scheme biases the training set by selecting samples with varying percentages of gradient magnitudes in order to increase the knowledge of

the learned content. We hypothesise that by drawing training samples with variance gradient magnitudes (i.e. in non-uniform proportions), the model will gain better learning performance than randomly uniform sample points due to targeting more informative groups.

The contributions of this chapter’s work focus on developing an algorithm for strategically selecting sample points to feed into a sequence prediction model in order to improve the overall model’s training and evaluation performance. Briefly, the main contributions are summarised as follows:

- A novel methodology of an offline data-driven sample selection scheme for training a many-to-one prediction model on a fixed size is proposed.
- The main intention is to reduce training volume and computation cost by extracting representative reduced training subsets while still achieving a remarkable compression ratio on large-scale medical datasets (16 bits).
- Evaluations compared to alternative data-driven sampling schemes (uniform, Gaussian, and sliced-based) have been examined to measure the effectiveness and quality of the chosen input samples on models’ learning outcomes.
- Experimentally indicates the generalisability and bit-reduction achievements of the predictive model when training it on our proposed subsampling scheme and comparing it to other state-of-the-art lossless alternatives.

The remainder of this chapter is organised as follows: Section 7.2 highlights the current research contributions in important sampling and subsampling within the deep learning domain. Section 7.3 explains the proposed sampling schemes and some alternative sampling strategies. Additional information on datasets, training settings, model architectures, and experimental results are provided in Section 7.4. The same section evaluates the compression results in (bpp) of the proposed models compared to the state-of-the-art lossless compression methods. The last Section 7.5, reports the outcomes and summarises the chapter.

7.2 Related Work

Subsampling large-scale data distributions tend to seek multiple objectives, including speeding up the training process, increasing convergence rate, reducing the computational cost, and reducing the error rate while solving the given task. Many strategies

have been developed to tackle some problems related to model capacity and the quality of the selected training sample points; namely, overfitting and underfitting [116]. Further objectives include enhancing the overall training performance and outcomes (i.e. training stability, time, generalisability, and convergence rate). When focusing on the reduction in dataset size, the advantages can further extend to include space and bandwidth savings as the data storage and communications are decreased. Such a feature would be notably desirable in the domains with bandwidth or power consumption limitations like multi-agents systems [38]. Another domain which significantly benefits from reducing the number of data points are the supervised learning tasks, specifically the human-annotated labelling. In such a task, the massive data size influences the given work as the labelling process is manually applied by human efforts, making it time-consuming and expensive [39]. By reducing the number of samples to more representative ones, the labelling workload would be notably reduced and faster to apply.

7.2.1 Contributions in Enhancing SGD Optimiser & Sampling Selection

A considerable amount of research contributions have been proposed to improve the learning performance when employing a Stochastic Gradient Descent (SGD) optimiser. The improvement involves many aspects, such as proposing variants of the SGD optimiser [179], improved batch-selection mechanisms [180], enhancing the quality of samples within mini-batches using an important sampling scheme to accelerate training and speed up convergence rate [181, 182], or introducing enhanced sampling of observations to reduce overfitting [183, 184]. Generally, the less variance a batch's gradients show, the more informative and further stabilised the training becomes, with a speedup gain in the training steps.

Korchi and Ghanou proposed a random sampling mechanism for drawing new training subsets to improve NN efficiency [183]. Their novel approach selects a new subset in each iteration while increasing the batches' diversity to avoid training instability problems (i.e. overfitting). The efficiency and generalisability of their data-driven method were empirically evaluated over three datasets (i.e. MNIST, CIFAR-10, and SVHN), showing better accuracy and training speed overall testing sets, although the models were trained only on 60% of the available data. A novel importance sampling for mini-batches was introduced by Csiba and Richtárik to solve supervised learning tasks [181]. The results of their method demonstrate improvement in training speed and reduction in stochastic variance compared to other uniform mini-

batching methods. The speedup factor was theoretically measured over both synthesis and real datasets, illustrating an order of magnitude improvement. Recent unbiased sampling that yields optimal batch variance for faster SGD convergence was proposed by Mariet [182]. In their batch selection method, they utilised learned distributions of the first and second-order marginals. Thus, the batches produce experimentally by their sampling method demonstrate lower-variance estimations that outperforms other batch-selection methods in convergence speed. Another optimised sampling method that picks representative and noiseless sample points specifically for training model solving One-Class Classification task was presented by Hadjadji, and Chibani [184]. Ioannou et al. [180] developed a novel biased batch-selection method, where more complex samples are selectivity added into training batches at each epoch. This biased sampling mechanism allows the deep neural network to converge faster with fewer training steps and better generalisation. The evaluations were conducted over four datasets (i.e. Boston housing, MNIST, CIFAR10, and CIFAR100), solving separate regression and classification tasks, respectively. Experimental results demonstrate accelerated convergence speed by up to 50% over the chosen datasets.

7.2.2 Contributions in Sampling Quality & Scoring Metrics

Other enhancements include modulating minibatch samples selections, producing important score metrics for determining training samples’ quality based on their impact on learning performance, investigating dataset characteristics (i.e. diversity and redundancy), [38–42].

Subsampling representatives and fewer data points yet sufficient for training form a significant challenge in deep learning domains, regardless of the tasks to be solved. The work proposed by Katharopoulos and Fleuret [40] demonstrated that not all samples have equal importance; in many cases, training can be efficiently ended with smaller training steps if representative and more informative data points are chosen. Their work focused on accelerating the training by employing an importance sampling scheme, which reduces the gradient variance while still picking samples with significant updating impacts on the NN parameters. An empirical study was proposed by Vodrhalli et al. [42], which investigates the quality and properties of some well-known datasets such as CIFAR-10/100, ImageNet and MNIST in terms of redundancy and diversity of samples. Similar to our work, this study utilises the gradient magnitude as their primary scoring metric. However, they only focus on analysing models’ performance trained only on the four image datasets. Their conclusion includes that some sample images are more important than others, given

evidence of model generalisability, which was one of the potential backbones for our proposed hypothesis. Another work that closely overlaps with our contribution but focuses on solving language modelling tasks was proposed by Fernandez, and Downey [41]. They seek to increase the learning outcomes while accelerating the learning process of the RNN language model by selecting more informative samples and complex sentences as training samples. Their results demonstrate that training a model on their weighted sampling strategy effectively outperforms the random sampling approach with similar sample size. Defining a compact but representative subset with minimum cardinality out of an original massive dataset has been experimentally investigated by Ghadikolaei et al. [38]. A demonstration of their proposed sampling algorithm for extracting representative samples was analytically and synthetically being evaluated. The algorithm expresses that the size of training samples can be reduced without much drop impact on the learning outcomes and performance. A diversity-driven sampling algorithm with boundary balancing constraints for solving classification tasks was proposed by Ramalingam et al. [39]. Their novel subset selection framework boosts performance gain while training using fewer training samples than other traditional deep-learning models. Experimental evaluation was carried out over four standard image classification datasets, including CIFAR-10, CIFAR-100, CIFAR-100-LT, and ImageNet.

Compared to the current literature, we proposed a novel subsampling scheme based on voxel’s gradient magnitude to learn domain-specific data distributions in a way that allows a many-to-one sequence prediction model to maximise its gained knowledge. We study the impacts of the chosen training subset generated by our offline data-driven importance sampling on the overall model’s compression performance. The sampling methods vary in homogeneous and heterogeneous regions scales and settings for threshold values. Many options have been considered based on the frequency distributions of voxels’ gradient magnitudes across volumes belonging to different datasets. Also, compression to other popular alternative strategies such as random, Gaussian, and slice-based sampling was presented. Furthermore, evaluation of the compression performance, as opposed to the state-of-the-art lossless methods, was highlighted.

7.3 Methodology

7.3.1 Problem Description

Given a data distribution defined over $\mathcal{P} \subset R^d$, the aim is to learn the distribution and gain knowledge about the specific population \mathcal{P} by selecting representative portions (i.e. a training subset $\mathcal{D} \subset \mathcal{P}$). The selection process of observations is known as sampling, while the total number of observations is described as sample size N . A subsampling strategy involves picking procedures when collecting datapoints, which can be classified into random and non-random schemes (e.g. stratified, clustered, or systematic). The common purpose is to collect sufficient data that explicitly represent the whole population. A sample is defined to be representative based on how well it describes the population or some of its characteristics. Thus, a strongly biased sampling scheme could systematically influence a study's outcomes as it may not reliably represent all individuals within a population. Generally, bias is not usually recommended in sampling as it may produce incorrect inferences about a population due to uniform selections. A selection is defined as biased if there are variations in the samples' selection likelihoods [185].

Selecting training samples is crucial and challenging in a deep learning context as it influences the overall learning process and stability. We formulated the data compression as a sequential prediction problem and solved it as a classical regression task. Our LSTM model is expected to learn a differentiable mapping function $f: \mathcal{X} \rightarrow \mathcal{Y}$ while adjusting the model's parameters θ through the back-propagation process given a training dataset \mathcal{D} . Where \mathcal{X} represents an input space, and \mathcal{Y} represents an output space. A training dataset \mathcal{D} consists of finite input-output pairs of samples with N sample size, whereby $\mathcal{D} = \{(X_i, y_i)\}_{i \in N}$, $X_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, and $\hat{y}_i = f(X_i)$. Each X_i consists of a sequence of multiple voxels' intensity values (i.e. $X_i = \{x_0, x_1, x_2, \dots, x_{l-1}\}$ with a fixed length l) surrounding a ground-truth target voxel known as y_i as illustrated in Fig. 7.1. As our LSTM model is fed with chains of input voxels' values X_i while predicting only a single value y_i as an output, thus it is known as a many-to-one sequence prediction model. During training, a minimisation of the loss function $\mathcal{L}(\hat{y}_i, y_i)$ is sought to reduce the distance between prediction \hat{y}_i and ground truth values y_i while updating the model's parameters θ . When evaluating the pre-trained model, a computation of the residual/prediction error E is applied, whereby the difference between prediction and actual value at each voxel location within the 3D volume is measured by $E_i = y_i - \hat{y}_i$. This volumetric residual E is reduced losslessly to a lower bit rate with an arithmetic coder on the encoder side.

When decompressing, a reversible decoding operation is applied at the receiver side by the arithmetic decoder to recover volumetric error E . Our LSTM model is then involved, whereby the \hat{y}_i voxels' values are progressively predicted and summed to their correspondence residuals values E_i in order to reconstruct the original volume (see Fig. 6.2 in chapter 6 for more details).

7.3.2 Alternative Sampling Strategies

The number of voxels included within some medical imaging datasets can be on a scale of billions, making training on all samples computational costly and impractical. The task of the many-to-one sequence prediction model seeks to learn a mapping from the input sequences to the target output through a set of training samples. Such a supervised learning process is affected by the quality of the selected sample points, especially on massive scale training datasets. The goal is to adaptively subsample data to reduce the number of observations required to train a model. The challenging question is whether it is possible to select a smaller set of critical data points that are more informative and representative to reduce training steps while not negatively influencing the overall learning performance. An illustration of the various sampling acquisition strategies applied in this chapter is proposed in Fig. 7.1, including *a*) Slice-based Sampling, *b*) Random (Uniform) Sampling, *c*) Gradient-based Sampling, and *d*) Gaussian Sampling.

7.3.2.1 Random Sampling

A random sampling of observations implies that all volumes' voxels have an equal probability of being selected. Moreover, uniformly subsampling voxels across multiple volumes is a simplistic and fair sampling technique with no biasing, making generalisations about the whole population, as demonstrated in Fig. 7.1 *b*). One of its main advantages is that it does not need prior knowledge about the data being collected. Moreover, it is one of the most popular sampling techniques as it applies an equitable procedure with no biasing when picking points.

7.3.2.2 Gaussian Sampling

Although in statistics, it is usually not recommended to apply a densely biased sampling scheme, in this chapter, we study the effect of a 3D Gaussian sampling on the overall model's learning performance. The intention is to utilise the domain knowledge about the nature of 3D medical scans, which usually locates the patient's

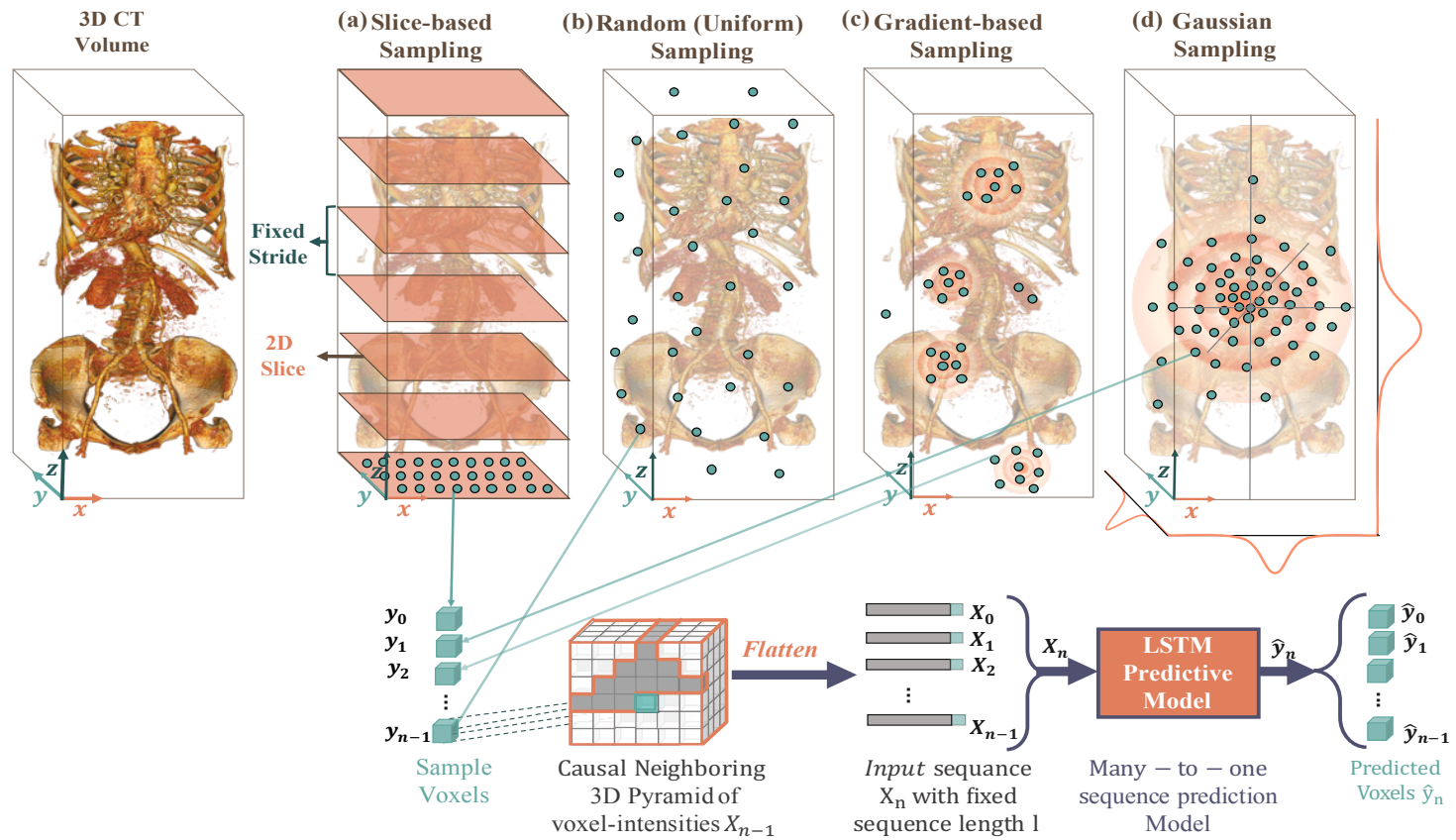


Figure 7.1: The sampling acquisition strategies applied in this chapter, including *a*) Slice-based Sampling, *b*) Random (Uniform) Sampling, *c*) Gradient-based Sampling, and *d*) Gaussian Sampling.

body in the middle of the image surrounded by air. Therefore, we hypothesise that our sequence prediction model may enhance the performance of the learned mapping function by biasing training samples toward the centre of a volume (i.e. more detailed body materials and tissues). In 3D Gaussian sampling, voxels from mid-slices would have higher probabilities for being selected than voxels at edges as shown in Fig. 7.1 *d*) (i.e. by intending to pick voxels from the body materials in the middle rather than air at the edge). Moreover, within our experiment, the sampling’s mean μ and standard deviation σ will vary based on the sampled volumes’ dimensions.

7.3.2.3 Slice-based Sampling

A slice-based sampling is a less random scheme but a more systemic approach in drawing samples from a population. This method can be utilised on high-dimensional data distribution such as 3D medical images or videos. All samples belonging to a cross-sectional slice or frame would be included in the dataset \mathcal{D} . The process of selecting all samples of a cross-section will repeatedly being applied to a volume’s z-axis after a fixed sampling interval (i.e. every n^{th} stride), as illustrated in Fig. 7.1 *a*). The intention is to thoroughly sample individual cross-sections region while intending to improve compression performance by learning multiple complete 2D slice distributions across the z-axis. This sampling approach has demonstrated effectiveness, and generalisation across many datasets through our findings in previous chapters 4, and 6 as it was comparable to other sampling schemes, namely, uniform sampling 6.4.6.

7.3.3 Proposed Sampling Method

7.3.3.1 Background

A gradient of a multivariate function f (e.g scalar data $f : R^n \rightarrow R^n$) is the first-derivative indicated as ∇f demonstrated in Eq 7.1, which is a vector representing the direction to the most significant change at a point p (i.e. directional change). At a particular point p in a scalar field, the gradient can be described as the slope of a tangent line, the instantaneous rate of change, and the direction to the steepness change (i.e. increasing or decreasing). At the same time, the gradient’s magnitude corresponds to the rate of change in that direction. The first-derivative f' and the second-derivative f'' of any function are considered as data-driven data features commonly involved in many fields and applications, including image processing and medical imaging domains. Such applications are applied to data with different dimensionalities. For instance,

in 2D space, edge detection or image segmentation are computed [186]. While in higher-dimensional spaces such as 3D spaces, mathematical computation of surface-based gradient norm for high-quality shading is employed [66]. More utilisations of the gradient’s magnitudes include 3D segmentation, 3D visualisation, the definition of the local change in the fields, and emphasising the material’s boundaries [1, 63, 66, 68].

$$\nabla f = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} \quad (7.1)$$

Where $\frac{\partial f}{\partial x}$ is the derivative with respect to x (i.e the gradient in the x direction) while $\frac{\partial f}{\partial y}$ is the derivative with respect to y (i.e. the gradient in the y direction) and $\frac{\partial f}{\partial z}$ is the derivative with respect to z (i.e. the gradient in the z direction).

In a volumetric medical dataset, a simplistic but popular procedure to approximate the gradient norm at each voxel location within the 3D grid is known as Central Differences [187, 188]. The main principle is to estimate the voxel normal by utilising the intensity values from its six local neighbourhoods, as computed in equation 7.3. The gradient magnitudes can then be determined by computing the vector’s length using equation 7.2.

$$\| f \| = \sqrt{g_z^2 + g_y^2 + g_x^2} \quad (7.2)$$

Where $\| f \|$ denotes the gradient’s magnitude, while g_x , g_y , and g_z represent the gradient’s vector components. This calculation can be precomputed and stored in a separate gradient volume to reduce the computational overhead, especially when handling large-scale high-resolution volumes.

$$\nabla f(v_x, v_y, v_z) = \left(\frac{v_{x+1} - v_{x-1}}{2h}, \frac{v_{y+1} - v_{y-1}}{2h}, \frac{v_{z+1} - v_{z-1}}{2h} \right)^T \quad (7.3)$$

Where $f(v_x, v_y, v_z)$ is the gradient approximation at voxel location (v_x, v_y, v_z) computed using Central Differences, while h denotes the distance between voxels in the 3D grid —(i.e. $h = 1$ for all volumes utilised within this chapter).

As gradient magnitude is a scalar quantity that emphasises the local rate of change at a particular voxel location, we selected to utilise it as a scoring mechanism within our importance sampling scheme. Determining gradient magnitudes provides intuitions about whether a voxel belongs to material, transition or boundaries between materials (i.e. a boundary is formed between high and low-intensity values). When the computed

vector’s length is zero, the corresponding voxel belongs to a homogenous region (e.g. materials), representing a critical point (i.e. local minima, local maxima, or a saddle point). However, a significant length result (i.e. high-gradient magnitudes) at the current voxel location indicates that the sample voxel belongs to a non-homogenous or heterogeneous region (e.g. materials’ boundaries). We empirically investigate the effect of choosing training samples belonging to homogenous or heterogeneous regions with different scales to learn various distributions sufficiently (see Figure 7.1 (c)). We gradually increased the proportion of sample points between the two areas and varied the magnitude’s threshold to find the best balance for enabling the LSTM model to gain the best compression performance.

7.3.3.2 Gradient-based Sampling

A gradient sampling method can be characterised as a type of cluster sampling, where voxels belonging to any volume are classified into different groups (i.e. homogeneous, and non-homogeneous regions) based on their gradient magnitude values. Then, random sampling is applied to collect the samples belonging to each cluster based on the specified important scores. A basic illustration of the methodology for this sampling scheme is shown in Fig. 7.1 c).

7.4 Experimental Results and Discussion

7.4.1 Performance Metrics

The bits-per-pixel (bpp) (Eq. 3.3.1) has been chosen to be the evaluation metric of the compression ratio obtained by all our LSTM models.

7.4.2 Dataset Details

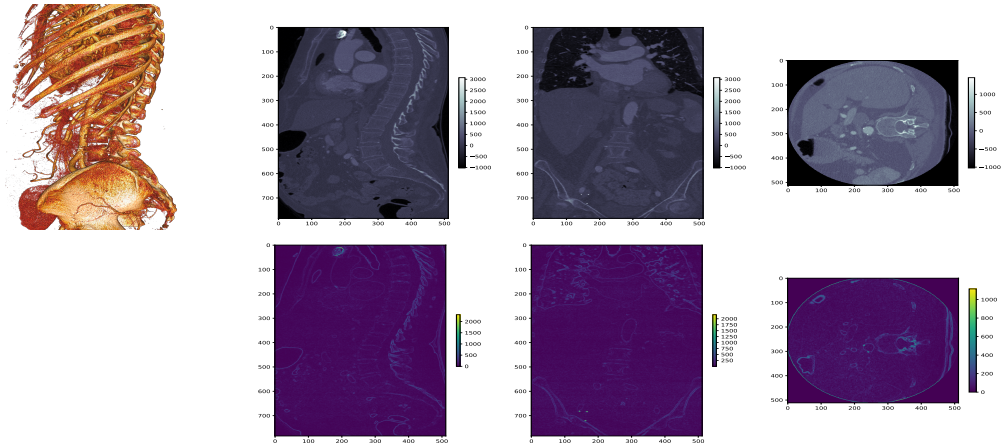
Dataset1 contains 12 high-resolution CT volumes with a total of 2.7 billion voxels available for training. Each volume represents a human Torso generated by a local hospital (private dataset) and stored as 16-bit grayscale (DICOM) files. A detailed overview of the volume specifications is shown in table 7.1 (for further details on the dataset specifications, refer to section 3.1.1). Visualisations of some sample volumes belong to **Dataset1** is presented in figure 7.2. Each subfigure illustrates 3D volume visualisation of the patient’s entire Torso with three orthogonal slice views (axial, sagittal and coronal) for colour mapping of slices’ intensity values (top row) and gradient magnitude values (bottom row). A validation procedure of 4-folds cross-

validations has curred out over this dataset (i.e. **Dataset1**), where each fold contains nine volumes belonging to the training set while three volumes are held for testing.

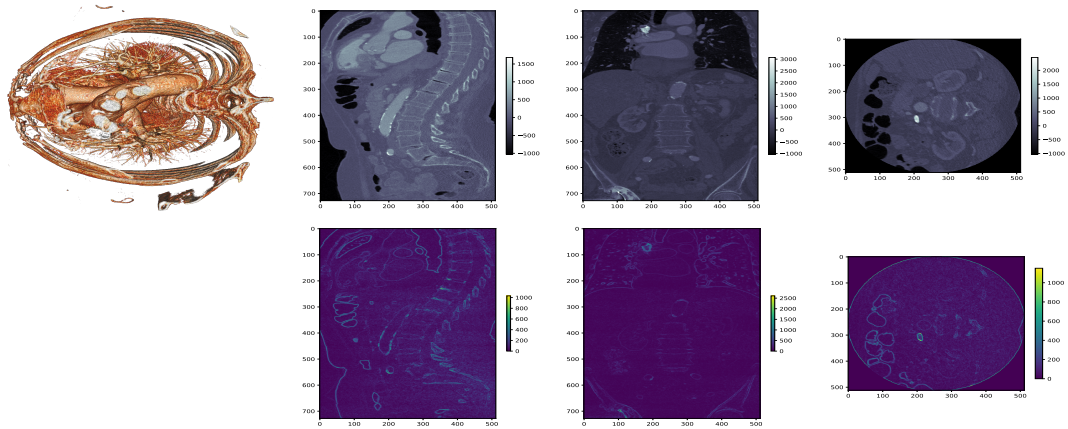
Volume ID	Resolution	Depth	Pixel Spacing, Slice Thickness	Minimum Value	Maximum Value
1	512×512	728	0.488, 0.488, 0.625	-1024	3071
2	512×512	1008	0.488, 0.488, 0.625	-1024	3071
3	512×512	784	0.488, 0.488, 0.625	-1024	3071
4	512×512	952	0.488, 0.488, 0.625	-1024	3071
5	512×512	784	0.488, 0.488, 0.625	-1024	3071
6	512×512	952	0.488, 0.488, 0.625	-1024	3071
7	512×512	952	0.488, 0.488, 0.625	-1024	3071
8	512×512	896	0.488, 0.488, 0.625	-1024	3071
9	512×512	840	0.488, 0.488, 0.625	-1024	3071
10	512×512	840	0.488, 0.488, 0.625	-1024	3071
11	512×512	728	0.488, 0.488, 0.625	-1024	3071
12	512×512	1008	0.488, 0.488, 0.625	-1024	3071

Table 7.1: **Dataset1** composed of 16 bit-depth medical images.

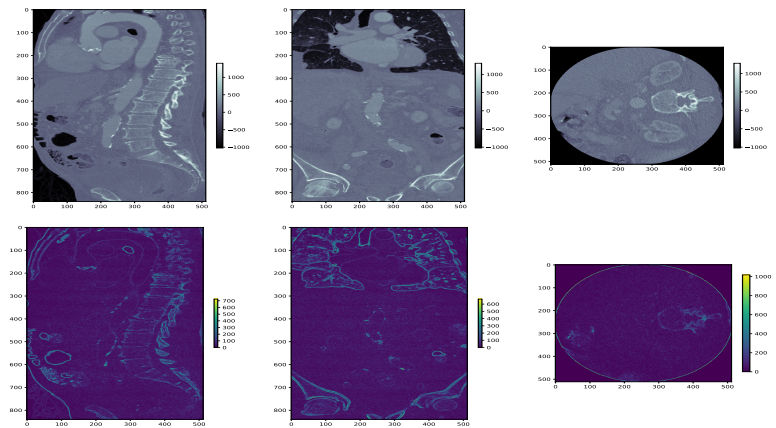
Another dataset used for evaluation purposes is a public set that contains around 377 million voxels in total, known as **Dataset2**. It contains 12 MRI (16 bit-depths) volumes of patients’ head and neck scans with $[.5, .5]$ pixel spacing, and $2mm$ slice thickness [16–18] (see section 3.1.3 for further details on the dataset specifications). Visualisations of some sample volumes belong to **Dataset2** is shown in Fig. 7.3. Each subfigure illustrates a patient’s head and neck with three orthogonal slice views (axial, sagittal and coronal) for colour mapping of slices’ intensity values (top row) and gradient magnitude values (bottom row). The performance of each of the proposed pre-trained models was measured in bpp 3.3.1 on this unseen out of domain dataset (see subsection 7.4.4.1 for more details). Another experiment was applied to this dataset with a validation procedure of 4-folds cross-validations, where each fold contains nine volumes belonging to the training set while three volumes are held for testing (see subsection 7.4.4.2 for more details).



a) Volume 1

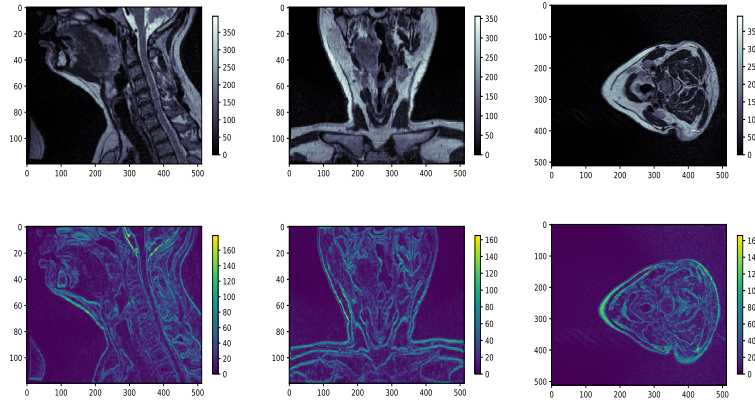


b) Volume 6

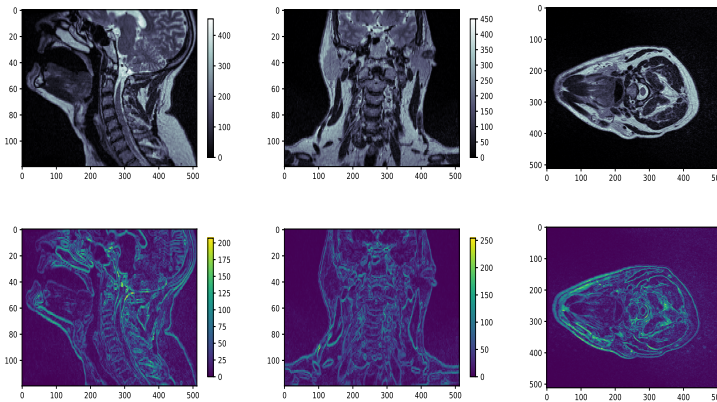


c) Volume 10

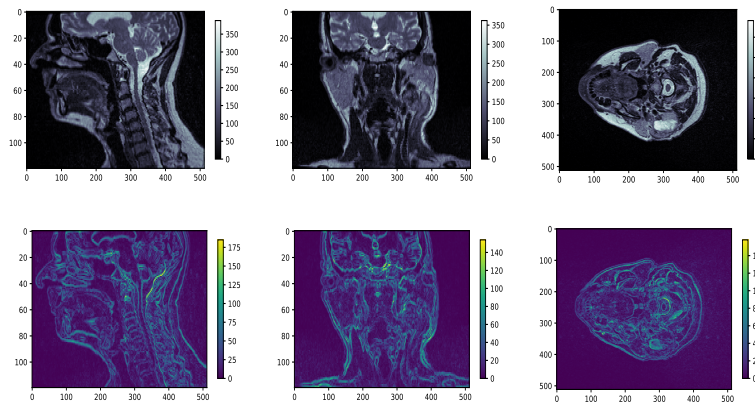
Figure 7.2: Visualisations of some (16 bit-depths) sample volumes belong to **Dataset1**. Each subfigure illustrates 3D volume visualisation of the patient's entire Torso with three orthogonal slice views (axial, sagittal and coronal) for colour mapping of slices' intensity values (top row) and gradient magnitude values (bottom row).



a) Volume 1



b) Volume 5



c) Volume 12

Figure 7.3: Visualisations of some (16 bit-depths) sample volumes belong to **Dataset2**. Each subfigure illustrates a patient's head and neck with three orthogonal slice views (axial, sagittal and coronal) for colour mapping of slices' intensity values (top row) and gradient magnitude values (bottom row).

7.4.3 Model Details

This subsection identifies the training settings and model details. To retain that the comparison between competing models is fair and convenient for all experiments, we employed the same RNN architecture in table 7.2 as the stander specification for all models used in this chapter. Although the size of this LSTM model is relatively compact (i.e. the storage size required for the model’s weights only is 276 KiloBytes (KB), while the complete model’s size (i.e. weights and training hyperparameters) is 810 Kilobytes (KB)), it has demonstrated that it has enough capacity to learn the 3D voxel correlation through our previous findings in [43, 44].

Layer	Number of Neurons	Activation Function Used
LSTM	128	Sigmoid and Tanh
Output	1	Linear

Table 7.2: The NN architecture used as the stander specification for all the proposed models.

For all the training applied in this chapter, we use Adam optimizer [177], with the following parameter settings $\beta_1 = 0.9, \beta_2 = 0.98$, a learning rate of $1e - 4$, and the training batch size is typically chosen to be 128. Each model was trained on a subset with a fixed sampling size for 60 epochs while minimising L_{joint} loss function (see section 3.2 for details).

7.4.4 Gradient-based Sampling

As the outcomes of our comprehensive study in chapter 6 demonstrated, we chose to train the models in this study on the best performer (i.e. sequences with the 3D-Pyramid (13x13,9x9,5x5,1) - N=175). The same sample size (i.e. 4.7M unique samples) was empirically employed in all experiments to keep the comparison among the different sampling schemes fair and convenient. Moreover, all chosen voxels drawn by different strategies are taken without replacements. We empirically evaluate the effectiveness, robustness, and generalisation for each distinct sampling method, including the various configurations within the gradient sampling scheme.

7.4.4.1 Gradient-based Sampling Applied to Dataset1

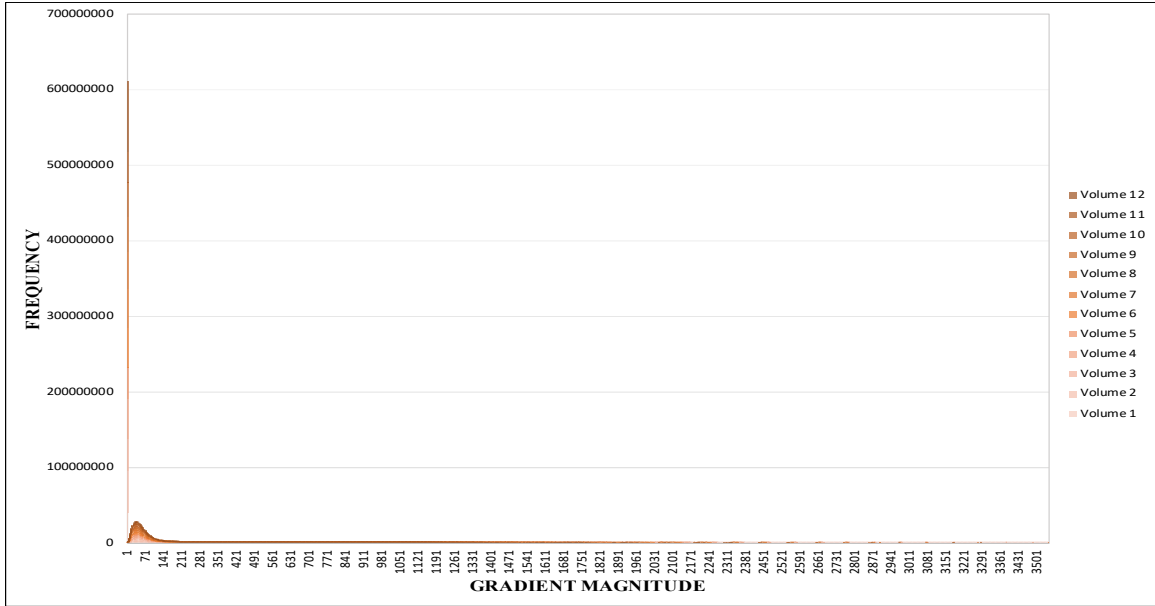
In the experiments, we gradually change the proportions for homogeneous and heterogeneous regions within the sampling buffer to find the balance that would lead to higher compression reduction. The study also includes changing the threshold rate

based on the frequency distribution of the gradient magnitude values. Figure 7.4 shows the gradients’ magnitudes histogram plots for both linear and logarithmic scales across **Dataset1** (sub-figures 7.4(a) and 7.4(b), respectively). From the frequency distribution, it is noticeable that a massive proportion of the voxels’ gradient normals are located within a homogeneous region (i.e. zero bin). Therefore, we sampled voxels within that area while varying the scale, in steps of 0% up to 60% and with threshold value of zero (i.e. Gradient Sampling 1, 2, 3, and 4 as demonstrated in Fig. 7.5). Sampling buffer with higher homogeneous divisions, for example, 80% and 100%, was empirically impractical to train-on, causing training instability problems (i.e. gradient explosion). Such a problem is expected when considering less diversity or the unbalanced size of duplicated samples.

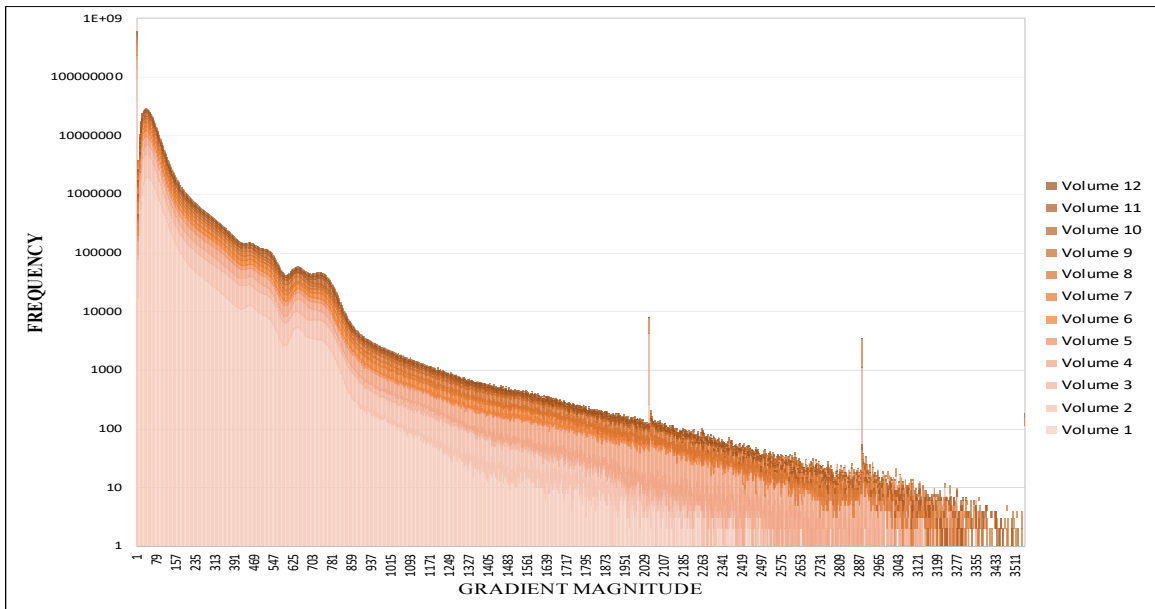
Sampling ID	Gradient Magnitude Threshold	Homogeneous Percentage	Non-Homogeneous Percentage	Configuration for Non-Homogeneous
1	0	0%	100%	-
2	0	20%	80%	-
3	0	40%	60%	-
4	0	60%	40%	-
5	0	20%	80%	(10, 100, 200, >)
6	0	20%	80%	(50, 100, 150, 200, 250, 300, 350, >)
7	10	40%	60%	-
8	100	40%	60%	-
9	100	40%	60%	No Air Intensity

Table 7.3: A summary overview of all the proposed gradient sampling schemes across **Dataset1**.

For Gradient Sampling 4 and 5, only 20% of the homogeneous region was included with a threshold of zero value while the rest 80% represent heterogeneous quarters. Based on the linear distribution of the gradient magnitude values, the bins with the higher frequencies have been massively sampled within each division. For instance, in Gradient Sampling 4, sample voxels within non-homogeneous regions with magnitude values (10, 100, 200, and other larger values) have a better chance of being selected up to 20% likelihood for each. Within Gradient Sampling 5, more magnitude distributions are chosen explicitly with a proportion of only 10% for each group, including (50, 100, 150, 200, 250, 300, 350, and other higher values) as shown in Fig. 7.5.



a) Gradient magnitudes histogram plot across **Dataset1**



b) Gradient magnitude (logarithmic scale) histograms across **Dataset1**

Figure 7.4: Gradient magnitudes histogram plots (linear & logarithmic scales) across **Dataset1**.

As in a scalar field, relatively homogeneous regions with low-gradient magnitudes would represent materials, while non-homogeneous areas have high-gradient magnitudes, indicating boundaries between materials. Thus, our experiments also include

likelihood patterns with different threshold values, including (10 and 100 magnitude values) for Gradient Sampling with 7, 8, and 9 IDs. The intention is to find a balance and diversity in voxel selections that benefit our sequence prediction model while learning the input-output mapping function. Through the examination of sampling subdivisions (i.e. homogenous and non-homogeneous classes), we found that proportions of 40% and 60% achieve a favourable correlation of compression performance and voxel selection. Thus, we further investigate defining an upper bound value for the homogeneous region that restricts elections within that limit. In Gradient Sampling 7, a threshold value of 10 was employed as the limit for the homogeneous group, while voxels with higher magnitude values have likelihoods for being chosen within the proportion of 60%. On the other hand, Gradient Sampling 8 has an upper bound of 100 for its homogeneous section. Similarly, Gradient Sampling 9 has a bound of 100 for its homogeneous section; however, we exclude voxels with air intensity value from being chosen in this sampling scheme. Although such sampling methodology can be relatively biased, the main objective is to massively focus all training samples on learning distributions of voxels from the body’s materials. A summary overview of all the proposed gradient sampling schemes across **Dataset1** is proposed in table 7.3.

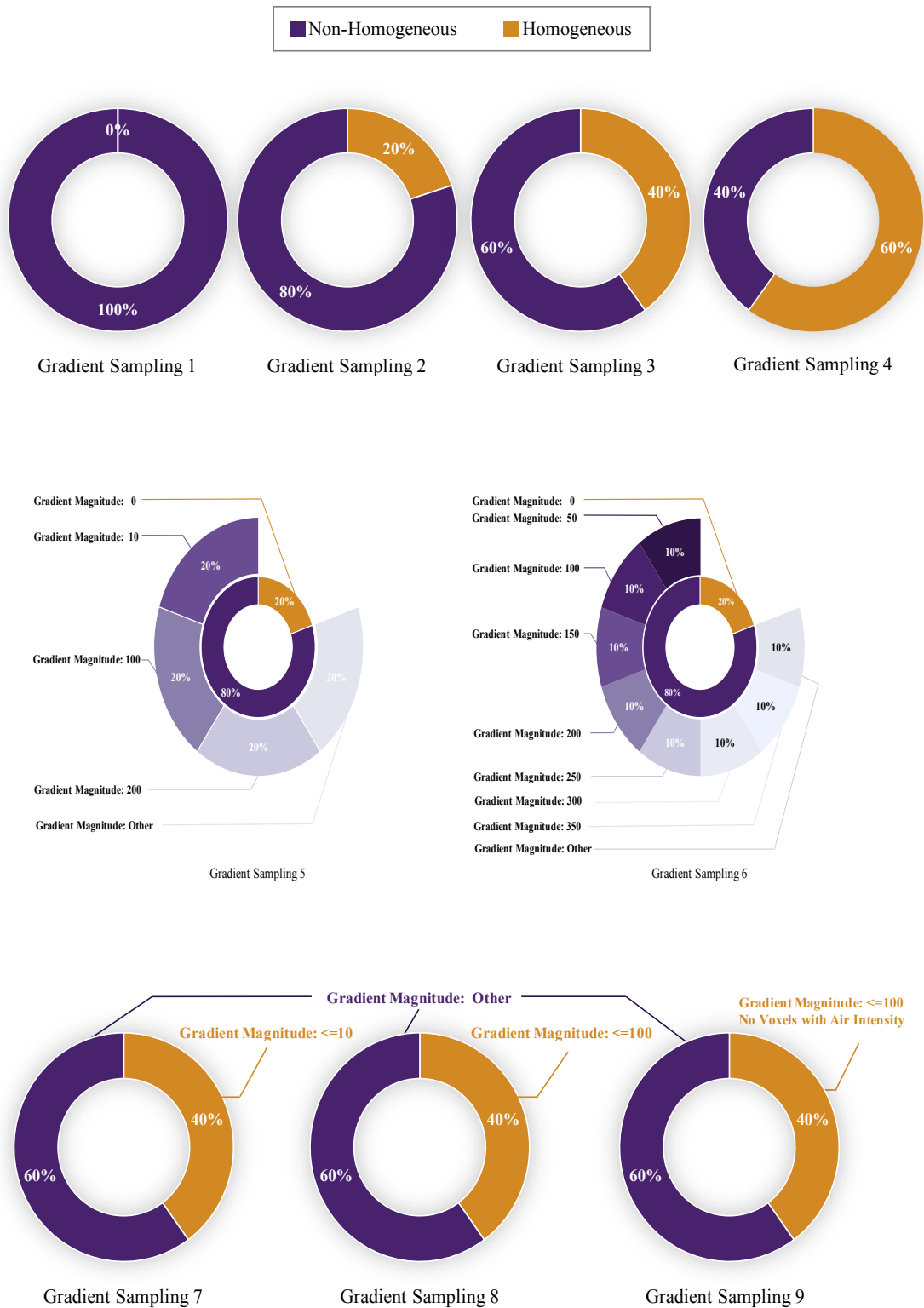


Figure 7.5: Visualisations of the proposed gradient sampling schemes. Each subfigure illustrates a subsampling approach utilised in drawing training samples.

		Slice-based Sampling	Random Sampling	Gaussian Sampling	Grad1 (th0_0%H_100%NH)	Grad2 (th0_20%H_80%NH)	Grad3 (th0_40%H_60%NH)	Grad4 (th0_60%H_40%NH)	Grad5 (th0_20%H_80%NH(10,100,200,>))	Grad6 (th0_20%H_80%NH(50,100,150,200,250,300,350,>))	Grad7 (th10_40%H_60%NH)	Grad8 (th100_40%H_60%NH)	Grad9 (th100_40%H_60%NH) - No Air
Fold 1	1	4.368	4.355	4.405	4.439	4.371	4.393	4.402	4.389	4.385	4.379	4.344	4.437
	2	4.331	4.313	4.378	4.388	4.334	4.35	4.353	4.355	4.342	4.339	4.302	4.385
	3	4.418	4.408	4.437	4.481	4.417	4.431	4.439	4.434	4.441	4.426	4.402	4.487
Fold 2	4	4.577	4.51	4.571	4.554	4.533	4.542	4.589	4.526	4.586	4.57	4.53	4.57
	5	4.175	4.143	4.193	4.196	4.171	4.177	4.214	4.16	4.246	4.202	4.169	4.216
	6	4.357	4.327	4.375	4.381	4.364	4.365	4.378	4.34	4.44	4.371	4.369	4.396
Fold 3	7	4.224	4.206	4.267	4.229	4.137	4.162	4.181	4.146	4.175	4.145	4.146	4.217
	8	4.392	4.42	4.47	4.434	4.345	4.362	4.377	4.342	4.379	4.351	4.349	4.442
	9	4.488	4.505	4.554	4.527	4.452	4.463	4.48	4.457	4.48	4.462	4.449	4.538
Fold 4	10	3.913	3.935	3.904	3.938	3.954	3.898	3.922	3.893	3.947	3.895	3.906	3.953
	11	4.343	4.355	4.325	4.352	4.375	4.321	4.341	4.301	4.355	4.313	4.316	4.354
	12	4.013	4.024	3.994	4.027	4.026	3.995	4.033	3.97	4.002	3.989	3.988	4.03
	avg	4.3	4.292	4.323	4.329	4.29	4.288	4.309	4.276	4.315	4.287	4.272	4.335
	std	0.19	0.181	0.206	0.194	0.177	0.193	0.19	0.195	0.191	0.196	0.186	0.194

Figure 7.6: Bits-per-pixel (bpp) for compressing **Dataset1** using models trained on different training subsets. The top labels specify the subsampling scheme specifications for drawing samples. 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. Cells are coloured from maximum compression 3.893 bpp (Purple) to the lowest compression 4.589 bpp (Brown).

An evaluation scheme of 4-fold cross-validation on **Dataset1** consisting of 12 high-resolution CT volumes were applied, whereas each model’s compression performance in bpp was reported in Figure 7.6. Each row represents the compression measurement in bpp over a single volume, while the combination of three volumes forms validation across a single fold – separated with a horizontal black line. Each column forms a specific model trained with a unique sampling scheme while the configurations are indicated on the top. The last row summarises the average bpp of all models through all volumes. Cells are coloured from maximum compression 3.893 bpp (Purple) to the lowest compression 4.589 bpp (Brown). The best compression result is in bold.

By examining the compression results of the alternative sampling methods, both random and slice-based sampling archives the same reduction with 4.29 on average

overall folds. Similarly, within the other subsampling approaches, Gradient Sampling Grad2 introduced the same reduction on average. Among the proposed gradient subsampling schemes, Grad8 with *threshold* = 100, a subdivision of 40% and 60% sections, achieves the best reduction with 0.45% space-saving on average as opposed to the standard random sampling method. On the other hand, the worst compression reduction was generated by Grad9, which is expected when strongly biasing the sampling to eliminate one of the data distributions (i.e. air intensity). However, the performance drop of this sampling scheme is only 1.45% compared to the best performer on average (i.e. Gradient Sampling Grad8).

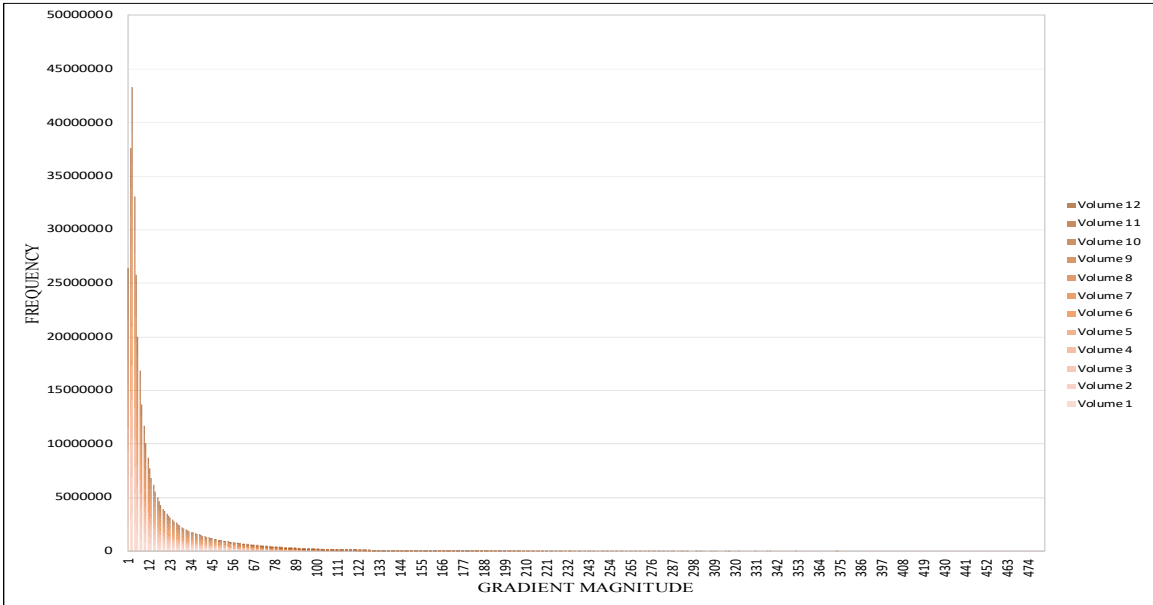
In figure 7.7, models pre-trained on **Dataset1** are now evaluated in bpp over an out of domain public set (i.e. **Dataset2**). This experiment measured the generalisation of each model trained on distinct subsampling schemes by evaluating its compression performance on unseen volumes (16 bit-depths). As each sampling approach was trained using a 4-fold cross-validation scheme, the evaluation of each of its trained models (i.e. four models per subsampling scheme) is grouped into corresponding folds separated by horizontal lines. To be precise, each column illustrates the reduction ratios of four models trained with the same sampling methodology across different volumes. However, each row presents the compression ratio of different models over a single volume except for the last row, which shows an estimation of average bpp achievements overall folds' means. Cells are coloured from maximum compression 2.9157 bpp (Purple) to the lowest compression 3.4342 bpp (Brown). The best compression result is in bold.

		Slice-based Sampling	Random Sampling	Gaussian Sampling	Grad1 (th0_0%H_100%NH)	Grad2 (th0_20%H_80%NH)	Grad3 (th0_40%H_60%NH)	Grad4 (th0_60%H_40%NH)	Grad5 (th0_20%H_80%NH(10,100,200,>))	Grad6 (th0_20%H_80%NH(50,100,150,200,250,300,350,>))	Grad7 (th10_40%H_60%NH)	Grad8 (th100_40%H_60%NH)	Grad9 (th100_40%H_60%NH) - No Air
Fold 1	1	3.078	3.071	3.117	3.088	3.082	3.131	3.19	3.067	3.102	3.085	3.074	3.081
	2	3.141	3.125	3.172	3.143	3.136	3.185	3.246	3.121	3.155	3.139	3.129	3.13
	3	3.125	3.118	3.166	3.137	3.127	3.18	3.241	3.116	3.148	3.132	3.119	3.124
	4	3.045	3.032	3.067	3.044	3.039	3.091	3.154	3.026	3.062	3.048	3.03	3.034
	5	3.309	3.301	3.37	3.322	3.316	3.377	3.434	3.297	3.329	3.316	3.302	3.304
	6	2.984	2.974	3.008	2.987	2.983	3.035	3.087	2.969	3.003	2.992	2.978	2.98
	7	3.132	3.118	3.156	3.128	3.125	3.174	3.237	3.108	3.146	3.13	3.118	3.118
	8	3.24	3.227	3.281	3.242	3.239	3.289	3.356	3.221	3.257	3.24	3.229	3.229
	9	3.162	3.149	3.197	3.161	3.155	3.206	3.271	3.139	3.176	3.159	3.149	3.147
	10	3.166	3.153	3.198	3.166	3.161	3.212	3.268	3.146	3.181	3.167	3.154	3.155
	11	3.048	3.035	3.075	3.053	3.046	3.099	3.152	3.033	3.075	3.052	3.038	3.047
	12	3.027	3.01	3.048	3.02	3.016	3.068	3.135	3.001	3.034	3.024	3.011	3.011
	avg		3.121	3.109	3.154	3.124	3.119	3.171	3.231	3.104	3.139	3.124	3.111
std		0.093	0.093	0.102	0.095	0.095	0.096	0.097	0.093	0.092	0.092	0.093	0.092
Fold 2	1	3.078	3.062	3.076	3.065	3.07	3.098	3.098	3.051	3.137	3.093	3.07	3.063
	2	3.136	3.121	3.132	3.124	3.122	3.156	3.159	3.11	3.193	3.149	3.124	3.123
	3	3.128	3.11	3.124	3.115	3.113	3.149	3.15	3.1	3.19	3.136	3.117	3.115
	4	3.046	3.025	3.035	3.028	3.027	3.066	3.062	3.013	3.102	3.057	3.03	3.025
	5	3.297	3.296	3.318	3.301	3.301	3.327	3.344	3.285	3.363	3.326	3.299	3.298
	6	2.983	2.966	2.978	2.972	2.971	3.007	3.004	2.954	3.046	2.999	2.976	2.966
	7	3.134	3.115	3.123	3.116	3.113	3.148	3.153	3.1	3.298	3.144	3.116	3.116
	8	3.239	3.225	3.234	3.225	3.223	3.257	3.265	3.211	3.214	3.252	3.226	3.224
	9	3.166	3.146	3.153	3.146	3.144	3.179	3.183	3.131	3.214	3.171	3.146	3.148
	10	3.163	3.147	3.159	3.15	3.149	3.184	3.189	3.137	3.222	3.176	3.151	3.152
	11	3.051	3.031	3.048	3.035	3.037	3.069	3.069	3.024	3.112	3.066	3.039	3.033
	12	3.033	3.003	3.011	3.007	3.003	3.045	3.046	2.989	3.087	3.032	3.008	3.006
	avg		3.121	3.104	3.116	3.107	3.106	3.141	3.143	3.092	3.182	3.133	3.108
std		0.09	0.094	0.096	0.094	0.094	0.091	0.096	0.094	0.091	0.093	0.092	0.095
Fold 3	1	3.068	3.103	3.096	3.081	3.06	3.084	3.096	3.043	3.088	3.068	3.049	3.08
	2	3.125	3.165	3.153	3.139	3.118	3.143	3.155	3.103	3.151	3.124	3.11	3.132
	3	3.117	3.156	3.144	3.126	3.106	3.13	3.145	3.093	3.139	3.114	3.096	3.123
	4	3.035	3.061	3.053	3.041	3.02	3.047	3.059	3.007	3.052	3.03	3.012	3.037
	5	3.29	3.35	3.34	3.314	3.291	3.32	3.338	3.273	3.321	3.295	3.283	3.307
	6	2.979	3.002	2.996	2.986	2.963	2.988	2.998	2.949	2.991	2.976	2.954	2.981
	7	3.117	3.152	3.141	3.129	3.11	3.137	3.148	3.093	3.14	3.119	3.103	3.124
	8	3.226	3.268	3.255	3.239	3.218	3.248	3.261	3.202	3.253	3.227	3.213	3.234
	9	3.149	3.185	3.175	3.157	3.138	3.17	3.18	3.124	3.172	3.149	3.134	3.152
	10	3.154	3.187	3.179	3.162	3.141	3.171	3.182	3.129	3.178	3.154	3.137	3.161
	11	3.041	3.069	3.062	3.049	3.021	3.053	3.065	3.016	3.062	3.037	3.026	3.052
	12	3.017	3.044	3.033	3.019	3.002	3.028	3.038	2.985	3.032	3.01	2.991	3.012
	avg		3.11	3.145	3.136	3.12	3.099	3.126	3.139	3.085	3.131	3.109	3.092
std		0.089	0.099	0.097	0.094	0.094	0.095	0.096	0.093	0.094	0.092	0.094	0.093
Fold 4	1	3.094	3.049	3.037	3.043	3.071	3.041	3.083	3.012	3.021	3.048	3.03	3.009
	2	3.152	3.11	3.094	3.105	3.131	3.102	3.147	3.075	3.078	3.107	3.092	3.067
	3	3.143	3.101	3.083	3.091	3.116	3.091	3.132	3.059	3.067	3.097	3.079	3.053
	4	3.063	3.016	2.998	3.006	3.03	3.01	3.048	2.977	2.984	3.017	2.998	2.973
	5	3.317	3.29	3.272	3.281	3.316	3.279	3.325	3.246	3.246	3.281	3.257	3.237
	6	3.003	2.957	2.944	2.947	2.973	2.949	2.988	2.918	2.93	2.957	2.941	2.916
	7	3.15	3.103	3.087	3.098	3.12	3.098	3.142	3.068	3.069	3.103	3.088	3.062
	8	3.252	3.213	3.197	3.207	3.234	3.206	3.252	3.178	3.174	3.209	3.192	3.167
	9	3.182	3.132	3.118	3.129	3.149	3.126	3.173	3.101	3.098	3.128	3.116	3.09
	10	3.181	3.135	3.126	3.132	3.153	3.128	3.175	3.103	3.103	3.133	3.118	3.093
	11	3.069	3.026	3.014	3.018	3.035	3.018	3.175	2.983	2.998	3.023	3.007	2.989
	12	3.046	2.998	2.975	2.988	3.011	2.989	3.029	2.957	2.961	2.994	2.98	2.951
	avg		3.138	3.094	3.079	3.087	3.112	3.087	3.139	3.056	3.061	3.091	3.075
std		0.089	0.094	0.094	0.095	0.097	0.094	0.094	0.095	0.09	0.092	0.091	0.091
All Folds	avg	3.123	3.113	3.121	3.11	3.109	3.131	3.163	3.084	3.128	3.114	3.097	3.096
	std	0.088	0.094	0.098	0.092	0.092	0.096	0.101	0.092	0.099	0.091	0.091	0.094

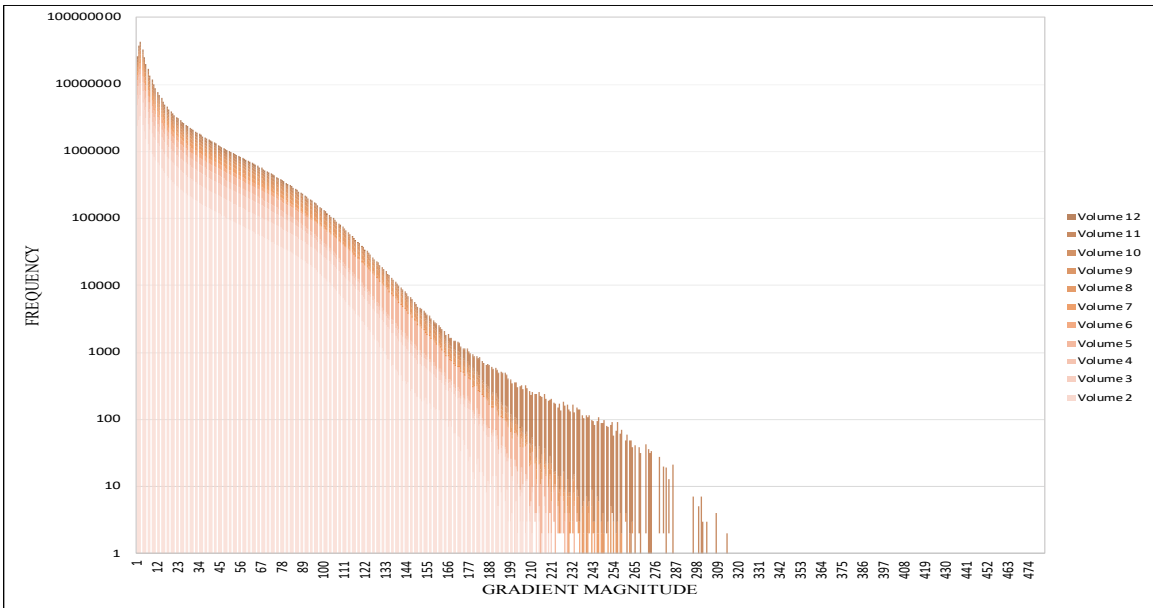
Figure 7.7: Compression performance in (bpp) for evaluating models' trained with different sampling strategies validated over **Dataset2**. Models pre-trained on distinct training subsets from **Dataset1** are evaluated on this set and grouped into corresponding folds separated by horizontal lines. The top labels specify the sampling configurations used when drawing training samples. Cells are coloured from maximum compression 2.9157 bpp (Purple) to the lowest compression 3.4342 bpp (Brown).

Across this particular dataset, it appears that the biased Gradient Sampling Grad9 achieves the best bpp reduction (i.e. 3.051 bpp) over all volumes in Fold4. By checking its mean over all folds, its average bpp ratios considered to be relatively comparable to the best performer. Although this subsampling excludes voxels with air intensity from the training buffer, it is still able to gain best reduction during the evaluation on this high-dynamic medical dataset. While looking into the compression reduction comprehensively, it is apparent that Gradient Sampling Grad5 produce the smallest bpp means overall folds with an average of 3.084 bpp. Precisely, this sampling scheme obtains equivalent bpp to Sampling Grad9 over Fold4 with 3.056 bpp. When only focusing on the validation results across Fold4, it appears that most of the proposed gradient sampling approaches gain comparable compression performance, including Sampling with the following IDs: 1, 3, 5, 6, 7, 8, and 9. Comparing the validation results of our best sampler to the alternative sampling schemes' (i.e. slice-based, random, and Gaussian) on the same fold, these methods are taking more noticeable storage spaces of 2.78%, 1.412%, and 0.9197%, respectively. The average mean of the alternative sampling schemes across all folds, yield relatively similar compression performance on average with almost 3.119 bpp. When comparing the bpp average reduction of the random sampling compared to the best performer (i.e. Gradient Sampling Grad6) over **Dataset2**, the gain in storage reach to almost 0.932%. On the other hand, the worst compression result on average on this dataset was accomplished by Grad4 with a mean of 3.163 bpp. Interestingly, the Gaussian sampling scheme, which performed the least reduction among all sampling methods over Dataset1, accomplished a better compression than slice-based and uniform sampling on this dataset. The evaluation results of Gaussian and Grad9 sampling schemes may indicate that by entirely biasing the training samples toward the tissues and body's materials, the learning-based coder may gain better generalisability, allowing better reduction for solving this particular task.

7.4.4.2 Gradient-based Sampling Applied to Dataset2



a) Gradient magnitudes histogram plot across **Dataset2**



b) Gradient magnitude (logarithmic scale) histograms across **Dataset2**

Figure 7.8: Gradient magnitudes histogram plots (linear & logarithmic scales across **Dataset2**).

Sampling ID	Gradient Magnitude Threshold	Homogeneous Percentage	Non-Homogeneous Percentage	Configuration for Non-Homogeneous
1	0	0%	100%	-
2	0	20%	80%	-
3	0	40%	60%	-
4	0	60%	40%	-
5	10	40%	60%	-
6	0	20%	80%	(1, 5, 10, >)

Table 7.4: A summary overview of all the proposed gradient sampling schemes across **Dataset2**.

As **Dataset2** has another modality (i.e. MRI) and represents different segments of the patient’s body (i.e. head and neck), the setting for the gradient sampling patterns may vary according to the voxels’ gradient magnitude frequency distributions. Figure 7.8(a) demonstrates the gradients’ magnitudes histogram plots for both linear and logarithmic scales across **Dataset2**. Examining the distribution would help choose the threshold value and the proportions for homogeneous and heterogeneous regions within the sampling buffer. Compared to Dataset1, both the voxels’ intensity and gradient magnitude frequency range is considerably small. Consequently, such a compact range would reduce the number of possibilities for gradient-based sampling. Based on the distribution, we introduce six main gradient-based sampling schemes. Starting by assigning the homogeneous division to 0% up to 60% while the threshold value was zero for Gradient Sampling 1 to 4.

Given the gradients’ magnitudes frequency plot, we focus on massively sampling voxels with high-frequency rates for the other two sampling schemes. For instance, Sampling Grad5 has a threshold value of 10 for its homogeneous group, while voxels with higher magnitude values have likelihoods for being chosen within the proportion of 60%. In Gradient Sampling Grad6, the threshold value of zero has a likelihood of only 20% while voxels’ with magnitude values (1, 5, 10, and other larger values) have a better chance of being included in the sampling buffer up to 20% for each. A summary overview of all the proposed gradient sampling schemes across **Dataset2** is proposed in table 7.4.

A similar experiment with an evaluation scheme of 4-fold cross-validation but on **Dataset2** was investigated in this section, whereas each model’s compression performance in bpp was described in Figure 7.9. Each fold demonstrates the validation over a group of three volumes, while each column denotes four models trained on subsets extracted with the same subsampling method over various volumes. The last row forms the mean overall folds, while the best compression result (bpp) is bold.

		Slice-based Sampling	Random Sampling	Gaussian Sampling	Grad1 (th0_0%h_100%NH)	Grad2 (th0_20%h_80%NH)	Grad3 (th0_40%h_60%NH)	Grad4 (th0_60%h_40%NH)	Grad5 (th10_40%h_60%NH)	Grad6 (th0_20%h_80%NH(1.5,1.0,->))
Fold 1	1	2.632	2.607	2.638	2.648	2.61	2.628	2.636	2.609	2.625
	2	2.687	2.668	2.696	2.708	2.673	2.687	2.697	2.67	2.687
	3	2.647	2.629	2.656	2.673	2.633	2.654	2.664	2.63	2.649
Fold 2	4	2.601	2.577	2.603	2.586	2.597	2.621	2.644	2.569	2.575
	5	2.795	2.778	2.804	2.79	2.804	2.832	2.862	2.766	2.777
	6	2.556	2.53	2.555	2.541	2.547	2.575	2.595	2.524	2.527
Fold 3	7	2.669	2.683	2.713	2.681	2.656	2.678	2.718	2.638	2.66
	8	2.752	2.775	2.805	2.774	2.745	2.77	2.811	2.729	2.751
	9	2.697	2.717	2.748	2.716	2.688	2.712	2.752	2.673	2.693
Fold 4	10	2.709	2.679	2.755	2.697	2.697	2.694	2.763	2.705	2.687
	11	2.665	2.611	2.694	2.639	2.632	2.629	2.69	2.641	2.622
	12	2.575	2.545	2.609	2.562	2.565	2.562	2.627	2.575	2.553
	avg	2.665	2.65	2.69	2.668	2.654	2.67	2.705	2.644	2.651
	std	0.07	0.082	0.08	0.078	0.074	0.077	0.08	0.07	0.075

Figure 7.9: Bits-per-pixel (bpp) for compressing **Dataset2** using models trained on different training subsets. The top labels specify the subsampling scheme specifications for drawing samples. 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. Cells are coloured from maximum compression 2.5238 bpp (Purple) to the lowest compression 2.8618 bpp (Brown).

Cells are coloured from maximum compression 2.5238 bpp (Purple) to the lowest compression 2.8618 bpp (Brown). The sampling schemes that perform the minimum compression reduction on **Dataset2** are Gaussian and Gradient-based sampling Grad4 with a bpp ratio of 2.705. On the other hand, models trained on subsets drawn by slice-based and random sampling accomplish comparable results with a compression ratio of approximately 2.658 bpp. Overall, the performance of models trained with sampling Grad5 improves the compression results and yields comparable results to the uniform sampling, which requires 0.223% more storage. A similar performance was produced by our other proposed gradient-based sampling with IDs 2 and 6.

7.4.5 Comparing with State-of-The-Art Lossless Compression Methods

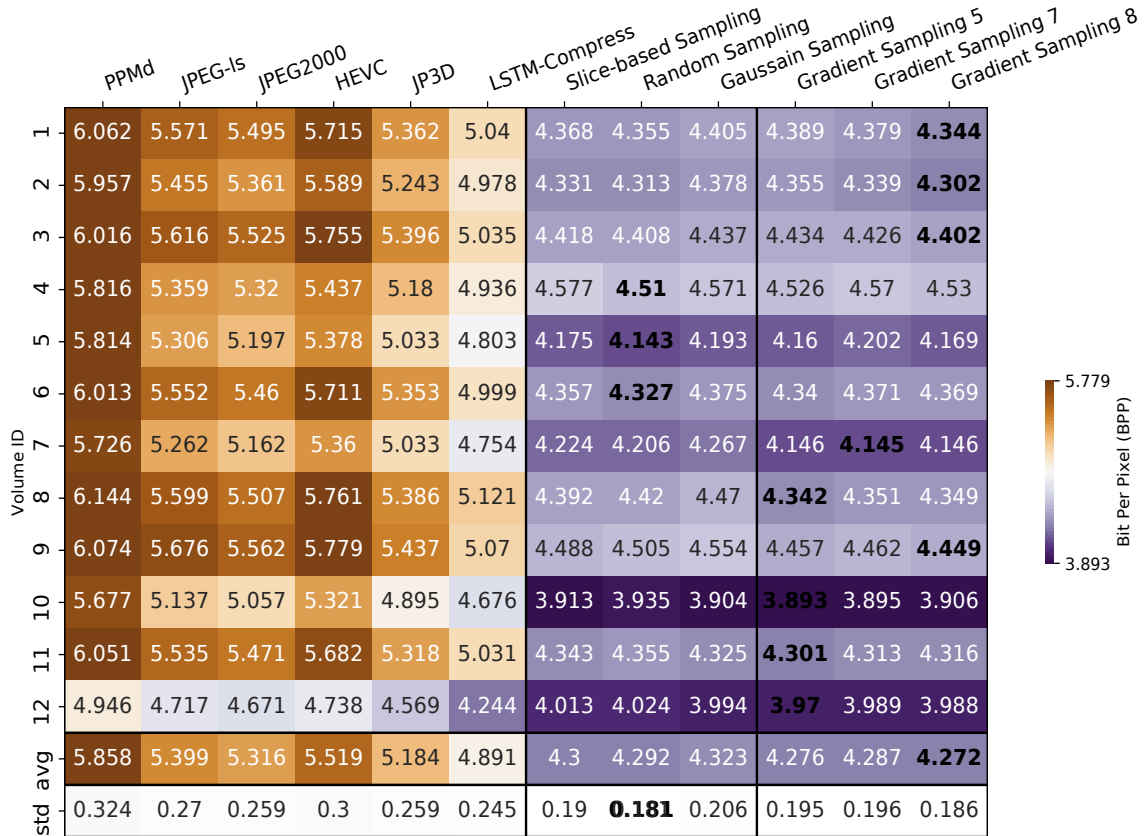


Figure 7.10: Illustrating the compression ratio in bpp of models trained with the some of proposed sampling schemes compared to the state-of-the-art lossless compression methods over **Dataset1** (16-bits volumes). Cells are highlighted from maximum compression 3.893 bpp (Purple) to minimum compression 5.779 bpp (Brown). The best compression result (bpp), is in bold.

	7Zip-PPMd	JPEG-ls	JPEG2000	HEVC	JP3D	lstm-compress	Slice-based Sampling	Random Sampling	Gaussian Sampling	Grad5 (th10_40%H_60%NH)	Grad6 (th0_20%H_80%NH(1.5,10,>))
1	4.245	3.941	3.558	4.079	3.707	3.424	2.632	2.607	2.638	2.609	2.625
2	4.246	3.966	3.57	4.068	3.702	3.455	2.687	2.668	2.696	2.67	2.687
3	4.281	3.969	3.572	4.096	3.709	3.458	2.647	2.629	2.656	2.63	2.649
4	4.133	3.875	3.487	3.99	3.621	3.379	2.601	2.577	2.603	2.569	2.575
5	4.52	4.158	3.757	4.273	3.885	3.654	2.795	2.778	2.804	2.766	2.777
6	4.101	3.827	3.438	3.96	3.579	3.338	2.556	2.53	2.555	2.524	2.527
7	4.163	3.899	3.516	4.025	3.662	3.428	2.669	2.683	2.713	2.638	2.66
8	4.327	4.04	3.644	4.137	3.776	3.536	2.752	2.775	2.805	2.729	2.751
9	4.201	3.922	3.534	4.034	3.679	3.459	2.697	2.717	2.748	2.673	2.693
10	4.257	3.969	3.581	4.101	3.719	3.483	2.709	2.679	2.755	2.705	2.687
11	4.153	3.879	3.503	4.029	3.655	3.409	2.665	2.611	2.694	2.641	2.622
12	4.073	3.803	3.42	3.899	3.553	3.326	2.575	2.545	2.609	2.575	2.553
std avg	4.225	3.937	3.548	4.058	3.687	3.446	2.665	2.65	2.69	2.644	2.651
std	0.12	0.096	0.091	0.095	0.088	0.088	0.07	0.082	0.08	0.07	0.075

Figure 7.11: Illustrating the compression ratio in bpp of all models trained with the proposed sampling schemes compared to the state-of-the-art lossless compression methods over **Dataset2** (16-bits volumes). Cells are highlighted from maximum compression 2.524 bpp (Purple) to minimum compression 4.273 bpp (Brown). The best compression result (bpp), is in bold.

This section measures the impact of our various data-driven sampling schemes on the compression performance compared to some well-known image and video coders. The state-of-the-art lossless compression methods include PPMd [175], JPEG-LS [173], JPEG2000 [174], HEVC [159,172], JP3D [174], and the deep learning method LSTM-Compress [146]. In this experiment, only models with the best performance over **Dataset1** and **Dataset2** are presented.

Based on the results of figure 7.10, it is noticeable that the maximum compression ratio is gained by our model trained with the proposed gradient sampling scheme (i.e. ID 8). When focusing on saving in space over **Dataset1**, compared to our Grad8, most of the classical compressors (i.e. PPMd, JPEG-ls, JPEG2000, HEVC, and JP3D) take up to 27%, 21%, 20%, 23%, and 18% **more** storage, respectively. Similarly, compared to the competing deep learning compressor (i.e. LSTM-Compress), this method needs

up to 13% **more** storage than our proposed model.

Over **Dataset2** in Fig. 7.11, compared to the best compressor Grad5, the state-of-the-art lossless compression methods (i.e. PPMd, JPEG-ls, JPEG2000, HEVC, and JP3D) need up to 37%, 33%, 25%, 35%, and 28% **more** storage. Moreover, Grad5 model demanded less storage requirements when compared against LSTM-Compress (i.e. a deep learning alternative codec), which need up to 23% **more** storage.

7.5 Conclusion and Further Work

We introduce multiple importance sampling approaches to enhance the quality and effectiveness of the training sample’s selections for LSTM compressor models. We observed that a more targeted subsample set was sufficient for learning the desired mapping function to solve the particular prediction-based task on the given 3D medical datasets. Although only 0.172% and 1.250% proportions of the large-scale medical datasets were utilised by each of the proposed sampling schemes, their compression results outperform alternative traditional compressors by a significant margin. We demonstrate that model trained on sampled extracted using gradient-based sampling scheme with (i.e. *threshold* = 10, 40.0% Homogeneous, and 60.0% Non-Homogeneous) yields the best bpp overall datasets with a better bpp reduction when compared to other models trained with random and biased training sets with similar sample size. By extracting a fixed number of representative training samples, a lower computation cost is obtained by leveraging parallelism over a limited proportion of the original massive dataset. Such a technique allows training the many-to-one LSTM model to predict the voxel values effectainly and more accurately given the reduced size of its residual values. Overall, the results of the gradient sampling are promising and intended to be a proof of concept given the benefits achieved in compression performance.

The following chapter will summarise the presented thesis, concluding all the principles and observations across all proposed techniques. It will also highlight some of the potential future works and research trends in the field of lossless compression of 3D medical images using deep learning methods.



Concluding Remarks

8.1 Summary of Contributions

In this PhD thesis, we addressed the problem of compressing high-resolution volumetric medical images losslessly using novel learning-based techniques. We formulated the compression problem as a sequence prediction problem and intended to solve it as a classical supervised regression task. To this end, we have provided contributions in the areas of lossless compression, 3D medical imaging, and deep learning.

Given the demand for ample storage by health care systems and its massive resolution scanned images, it is desirable to learn the spatial information and exploit data characteristics to reduce the amount of information required intelligently and efficiently. We initially present novel 3D localised sampling methods (cube-shaped and pyramid-shaped) in chapter 3 to detect voxels' spatial features from the underlying geometric structure of the medical images domain. Additionally, a novel encoder-decoder compression framework consisting of a neural network predictor model followed by an arithmetic coder was utilised. We also developed a joint loss function that combines the Mean Absolute Error (MAE) with the Pearson Correlation Coefficient (PCC). We found that incorporating these two losses for solving the regression task has a significant impact in enhancing the accuracy and stability of the training. The main intention is to investigate the effectiveness of learning a projection function leveraged by a neural network model. An evaluation of the learned mapping was investigated over various datasets while identifying its strengths and limitations. Although the model's size is only *7MB*, the compression achievements were optimistic and proved its compression efficacy compared to the alternative well-known standard lossless

compressors. The main limitation was related to the model’s generalisation. Given that our prediction model learns the spatial features at the voxels level, changes in the scanners’ quality or parameters would affect the overall compression performance. Therefore, the feasibility for this model to generalise on unseen datasets is practically accomplished when retaining a satisfactory performance.

In order to improve the model’s compression accuracy and generalisability, in chapter 4, we employed LSTM cells as our prediction-based model, leveraging its strength in memorising the sequences of 3D spatial voxels information. By utilising its gating mechanism, the LSTM block can exploit long dependencies within the input sequence, allowing better compression achievements while solving the given many-to-one prediction problem. A minimal yet sufficient model’s weights size (i.e. only 810 Kilobytes (KB)) achieves the best reduction while still retaining the bandwidth transmission potential limit when sending compressed representation with the decoder. The contribution also includes examining the effect on compression quality when training on samples drawn from images with different scanning parameters. Both robustness and generalisability were gained when picking training examples from different data distributions (i.e. images with varying pixel spacing and slice thickness). Compared to the other state-of-the-art lossless compressors, our MedZip models can reduce high-resolution 3D medical scans to higher bit reductions within a high degree of efficiency when predicting at a voxel level.

Chapter 5 examines the use of various causal neighbouring elements extracted from medical imagery on the learned mapping function acquired by a prediction-based compressor. We proposed the first comprehensive study on voxel-wise prediction using several input sampling schemes. Both compression ratio and time were measured for each proposed model trained on a district input sequence to highlight trade-offs and performance gains while defining the optimal sequence configuration. When focussing on the 3D spatial voxel’s association, the pyramid-shaped sequence demonstrated outstanding balances between compactness and representativity while capturing local correlations. The study contributions include leveraging parallelism and speeding up the decompression process by introducing a novel methodology that eliminates left voxels from sequences. Such a procedure results in a drop in compression ratio (i.e. only $\approx .2$ bpp) while considerably payback in speeding up decoder performance up to $37\times$. Our predictive codec has a noticeably small model size of only 810(KB) yet sufficient to outperform state-of-the-art codecs by a significant margin.

In chapter 6, a novel data-driven subsampling technique was proposed to decrease the number of training samples required while still gaining favourable compression

performance. The development of a data-driven important sampling method aids the learnability outcomes, while substantially reducing the computation cost and time needed compared to impractically training on entirely massive original data. Evaluation performance in bpp of the pre-trained models was chosen to be the primary measurement of the effectiveness of the proposed subsampling scheme. As our sequence prediction model exploits voxels' 3D local correlations, picking the voxels' gradient magnitude as a scoring metric for our subsampling method supports identifying spatially localized features. The study also empirically investigates setting distinctive gradient thresholds and selecting multi-scale homogeneous and heterogeneous segments to find the balances which would lead to better performance. When examining the compression achievements, we found that a set with ($threshold = 10$, 40.0% homogeneous, and 60.0% non-homogeneous) sampling configurations experimentally demonstrates having representative and sufficient sample variations across different datasets. Consequently, this importance sampling methodology demonstrates promising results that outperform not only state-of-the-art lossless compressors but also models trained on sets drawn by other sampling schemes, including uniform, slice-based, and Gaussian sampling methods.

Predictive Codec	Method Type	Number of Causal Neighborhood Utilised	Dimensions of the Causal Neighborhood Utilised	Compression Quality	Data Compression Domains	Highlights
NN-Based Predictive Model [43]	DL-based	3D Cubic and 3D Pyramid causal neighborhood	3D	Lossless	3D Medical Images (16 bit-depths) CT Scans	A novel encoder-decoder compression framework consisting of a neural network predictor model followed by an arithmetic coder was utilised for compressing volumetric medical images losslessly. Our NN-predictor utilises novel 3D localised sampling methods (cube-shaped and pyramid-shaped) to capture voxels' spatial features from the underlying geometric structure of the medical images domain. The main intention is to investigate the effectiveness of learning a projection function leveraged by a neural network model. The main limitation was related to the model's generalisation. Given that our prediction model learns the spatial features at the voxels level, changes in the scanners' quality or parameters would affect the overall compression performance.
MedZip Predictive Model [44]	DL-based	3D Cubic and 3D Pyramid causal neighborhood	3D	Lossless	3D Medical Images (16 bit-depths) CT and MRI Scans	A compression framework based on the LSTM predictor was introduced to code 3D medical images losslessly while improving the model's compression accuracy and generalisability. The intention of using the LSTM model as a 3D predictor is to leverage its strength in memorising sequence dependences over 3D spatial information across different volumes. Both robustness and generalisability were gained when picking training examples from different data distributions (i.e. images with varying pixel spacing and slice thickness). Compared to the other state-of-the-art lossless compressors, MedZip models can efficiently reduce large-scale medical volumes to higher bit reductions when predicting at voxel levels.
Comprehensive Sampling Strategies for Predictive Model [45]	DL-based	1D, 2D, and 3D (Cubic and Pyramid shaped) causal neighborhood	1D/2D/3D	Lossless	3D Medical Images (16 bit-depths) CT and MRI Scans	The first comprehensive study of a voxel-wise prediction model was introduced to find the best pattern that would lead to optimal compression quality and performance. Studying the use of various input features extracted from the causal medical distributions on the learned mapping function acquired by a prediction-based compressor. Both compression ratio and time were measured for each proposed model trained on a district input sequence to highlight trade-offs and performance gains while defining the optimal sequence configuration. The study contributions include leveraging parallelism and speeding up the decompression process by introducing a novel methodology that eliminates left voxels from sequences.
Gradient-based Importance Sampling for Predictive Model	DL-based	3D Pyramid causal neighborhood	3D	Lossless	3D Medical Images (16 bit-depths) CT and MRI Scans	A novel data-driven subsampling technique was proposed to decrease the number of training samples required while still gaining favourable compression performance. Our many-to-one prediction model exploits voxels' 3D local correlations, so picking the voxels' gradient magnitude as a scoring metric for our subsampling method supports identifying spatially localized features. The study also empirically investigates setting distinctive gradient thresholds and selecting multi-scale homogeneous and heterogeneous segments to find the balances which would lead to a better model's performance. The proposed sampling scheme demonstrates promising results that outperform not only state-of-the-art lossless compressors but also models trained on sets drawn by other sampling schemes, including uniform, slice-based, and Gaussian sampling methods.

Table 8.1: Overall summary table for the main contributions proposed in this PhD thesis.

8.2 Future Research Work

An essential step in any research contribution is to reflect on the current work’s limitations and future research trends to identify open questions, highlight potential enhancements, and support developing future novel alternative strategies. The work presented in this thesis can be extended in the following directions:

- The fundamental objective of our proposed work throughout this thesis is to compress 3D medical images losslessly. Thus, intending to evaluate our compressors on other potential high-dimensional data domains such as videos would be fascinating.
- Investigating and exploring the performance of utilising other varieties of neural network architectures as a backbone for solving the same compression task. Utilising different NN architectures would allow highlighting the strengths and limitations of each type over different data domains while examining the impact on learning performance and compression performance.
- Deeply analysing the generalisation across the learned mapping of spatially distinct 3D features (e.g. pixel spacing, and slice thickness), different scans’ settings, and quality, while intending to use this knowledge will help facilitate the development of more generalised and robust deep learning models. Other potential directions include studying a model’s generalisability across the same patient scanned with different scanners, scanning quality, or modalities (e.g. medical images of the same patient scanned in CT and MRI).
- Studying the effect of model size and weight sparsity on compression ratio and transmitting both compressed representation and decoder.
- While we select the gradient magnitude as our primary score metric for sub-sampling datapoints, enhancing the sampling methodology or combining it with other alternative metrics would be interesting to improve the outcomes.
- Further investigate automating the gradient-based sampling technique in choosing the threshold value, homogenous, and heterogeneous regions adaptively according to the underlying dataset characteristics. As in our proposed contribution, the threshold values and region ratios are manually assigned based on the data distribution; automating the selection would further expand the domain applications while independently adjusting to different dataset characteristics.

- Although the proposed importance sampling metric used for our offline data-driven sampling scheme proves its effectiveness, it would be desirable to leverage its strength in extracting samples immediately for in-loop training. An enhancement would be examining the possibility of extracting a new subset in each iteration while increasing the batch’s diversity and strategically targeting more informative training samples (e.g. including more complex samples into training batches at each epoch [180]).
- Investigate the potential for designing a novel End-to-End lossless compression framework that jointly learns to compress images/volumes bit-rate while optimising the overall model performance (e.g. [166]).
- Introduce a systematic survey and benchmarks to thoroughly summarise and compare learning-based lossless codecs.
- One of the common and non-trivial challenges related to data quality, specifically 3D medical images, is registration. This process tends to align the stack of 2D images to match the corresponding anatomical regions and spatial locations perfectly. The quality of such a task is crucial as the accuracy of the registration algorithm would influence the overall performance of any postprocessing or analysis techniques that would follow up. However, in the case of a lossless compression procedure, it is required that all the information remains intact as any modifications are intolerable. Although such a problem may affect the overall learning quality, applying the registration method within the lossless reduction is unallowable. Therefore, the data providers should apply such approach to enhance the volume quality and accuracy before any posterior operations can be applied.

Overall, the potential of deep learning within the fields of data compression and medical analysis is massive. This field has numerous open opportunities and gaps in science that could be filled with new and novel contributions. Developing learning-based schemes within such an emerging field would benefit a wide range of applications and communities, principally when focusing on lossless compression quality.

It is my hope that the findings presented in this thesis will lay the foundations of a new and exciting field of study, combining modern deep learning and lossless compression techniques in principled and practical ways.

REFERENCES

- [1] A. Telea, *Data Visualization: Principles and Practice, Second Edition*. Taylor & Francis, 2014. [Online]. Available: <https://books.google.co.uk/books?id=2WoLBAAAQBAJ>
- [2] D. Tao, S. Di, Z. Chen, and F. Cappello, “Exploration of pattern-matching techniques for lossy compression on cosmology simulation data sets,” in *International Conference on High Performance Computing*. Springer, 2017, pp. 43–54.
- [3] M. Goyal, K. Tatwawadi, S. Chandak, and I. Ochoa, “Dzip: Improved general-purpose lossless compression based on novel neural network modeling,” in *Data Compression Conference*, A. Bilgin, M. W. Marcellin, J. Serra-Sagrìstà, and J. A. Storer, Eds. IEEE, 2020, p. 372. [Online]. Available: <https://doi.org/10.1109/DCC47342.2020.00065>
- [4] S. Cao, C. Wu, and P. Krähenbühl, “Lossless image compression through super-resolution,” *CoRR*, vol. abs/2004.02872, 2020. [Online]. Available: <https://arxiv.org/abs/2004.02872>
- [5] T. Kim, H. M. Kim, P. Tsai, and T. Acharya, “Memory efficient progressive rate-distortion algorithm for JPEG 2000,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 181–187, 2005. [Online]. Available: [https://doi.org/10.1109/TCSVT.2004.839970\(410\)1](https://doi.org/10.1109/TCSVT.2004.839970(410)1)
- [6] M. A. Rahman, M. Hamada, and J. Shin, “The impact of state-of-the-art techniques for lossless still image compression,” *Electronics*, vol. 10, no. 3, 2021. [Online]. Available: <https://www.mdpi.com/2079-9292/10/3/360>
- [7] P. Schelkens, A. Munteanu, A. Tzannes, and C. M. Brislawn, “JPEG2000. part 10. volumetric data encoding,” in *International Symposium on Circuits and Systems*. IEEE, 2006, pp. 4–3877. [Online]. Available: <https://doi.org/10.1109/ISCAS.2006.1693474>

- [8] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, 2000. [Online]. Available: <https://doi.org/10.1109/83.855427>
- [9] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012. [Online]. Available: <https://doi.org/10.1109/TCSVT.2012.2221191>
- [10] L. F. R. Lucas, N. M. M. Rodrigues, L. A. da Silva Cruz, and S. M. M. de Faria, "Lossless compression of medical images using 3-d predictors," *IEEE Trans. Medical Imaging*, vol. 36, no. 11, pp. 2250–2260, 2017. [Online]. Available: <https://doi.org/10.1109/TMI.2017.2714640>
- [11] A. Kalra, "Chapter 9 - developing fe human models from medical images," in *Basic Finite Element Method as Applied to Injury Biomechanics*, K.-H. Yang, Ed. Academic Press, 2018, pp. 389–415. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B978012809831800009X>
- [12] D. Dance, S. Christofides, A. Maidment, I. McLean, and K. Ng, *Diagnostic radiology physics: A handbook for teachers and students. Endorsed by: American Association of Physicists in Medicine, Asia-Oceania Federation of Organizations for Medical Physics, European Federation of Organisations for Medical Physics*, ser. Non-serial Publications. Vienna: International Atomic Energy Agency, 2014. [Online]. Available: <https://www.iaea.org/publications/8841/diagnostic-radiology-physics>
- [13] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1309–1324, 2000. [Online]. Available: <https://doi.org/10.1109/83.855427>
- [14] B. Zhao, L. P. James, C. S. Moskowitz, P. Guo, M. S. Ginsberg, R. A. Lefkowitz, Y. Qin, G. J. Riely, M. G. Kris, and L. H. Schwartz, "Evaluating variability in tumor measurements from same-day repeat CT scans of patients with non-small cell lung cancer," *Radiology*, 2009.
- [15] B. Zhao, L. H. Schwartz, and M. G. Kris, "Data from rider lung CT. The Cancer Imaging Archive," (*TCIA*), 2015.

- [16] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle, L. Tarbox, and F. Prior, “The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository,” *Journal of Digital Imaging*, vol. 26, no. 6, 2013. [Online]. Available: <https://doi.org/10.1007/s10278-013-9622-7>
- [17] C. Cardenas, A. Mohamed, G. Sharp, M. Gooding, H. Veeraraghavan, and J. Yang, “The cancer imaging archive.” *Data from AAPM RT-MAC Grand Challenge*, 2019. [Online]. Available: <https://doi.org/10.7937/tcia.2019.bcfjqfqb>
- [18] C. E. Cardenas, A. S. R. Mohamed, J. Yang, M. Gooding, H. Veeraraghavan, J. Kalpathy-Cramer, S. P. Ng, Y. Ding, J. Wang, S. Y. Lai, C. D. Fuller, and G. Sharp, “Head and neck cancer patient images for determining auto-segmentation accuracy in T2-weighted magnetic resonance imaging through expert manual segmentations,” *Medical Physics*, vol. 47, no. 5, pp. 2317–2322, 2020. [Online]. Available: <https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.13942>
- [19] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez, “A survey on deep learning in medical image analysis,” *Medical Image Anal.*, vol. 42, pp. 60–88, 2017. [Online]. Available: <https://doi.org/10.1016/j.media.2017.07.005>
- [20] S. Dixon, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Oct. 2013.
- [21] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Nov. 2014.
- [22] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Oct. 2015.
- [23] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Oct. 2016.
- [24] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Nov. 2017.
- [25] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Sep. 2018.
- [26] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Sep. 2019.

- [27] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Oct. 2020.
- [28] —, “Diagnostic imaging dataset statistical release,” NHS England, UK, Tech. Rep., Nov. 2021.
- [29] K. Anagnostakos, Y. Knafo, F. Houfani, B. Zaharia, F. Egrise, I. Clerc-Urmès, and D. Mainard, “Value of 3D preoperative planning for primary total hip arthroplasty based on biplanar weightbearing radiographs,” *BioMed Research International*, vol. 2019, p. 1932191, 2019. [Online]. Available: <https://doi.org/10.1155/2019/1932191>
- [30] D. A. Koff and H. Shulman, “An overview of digital compression of medical images: can we use lossy image compression in radiology?” *Canadian Association of Radiologists Journal*, vol. 57, no. 4, p. 211, 2006.
- [31] G. Patidar, S. Kumar, and D. Kumar, “A review on medical image data compression techniques,” in *International Conference on Data, Engineering and Applications*. IEEE, 2020, pp. 1–6.
- [32] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, “Practical full resolution learned lossless image compression,” in *IEEE Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2019, pp. 10 629–10 638. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Mentzer_Practical_Full_Resolution_Learned_Lossless_Image_Compression_CVPR_2019_paper.html
- [33] M. C. de Oliveira, L. G. R. Martins, H. C. Jung, N. D. G. Júnior, R. C. da Silva, E. Peixoto, B. Macchiavello, E. M. Hung, V. Testoni, and P. G. Freitas, “Learning-based end-to-end video compression using predictive coding,” in *SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE, 2021, pp. 160–167. [Online]. Available: <https://doi.org/10.1109/SIBGRAPI54419.2021.00030>
- [34] G. Flamich, M. Havasi, and J. M. Hernández-Lobato, “Compression without quantization,” 2019.
- [35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017. [Online]. Available: <http://doi.acm.org/10.1145/3065386>
- [36] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning*

- Representations*, Y. Bengio and Y. LeCun, Eds., 2015, p. arXiv:1409.1556. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [37] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2016, pp. 770–778. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.90>
- [38] H. S. Ghadikolaei, H. G. Ghauch, C. Fischione, and M. Skoglund, “Learning and data selection in big datasets,” in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 2191–2200. [Online]. Available: <http://proceedings.mlr.press/v97/ghadikolaei19a.html>
- [39] S. Ramalingam, D. Glasner, K. Patel, R. Vemulapalli, S. Jayasumana, and S. Kumar, “Balancing constraints and submodularity in data subset selection,” *CoRR*, vol. abs/2104.12835, 2021. [Online]. Available: <https://arxiv.org/abs/2104.12835>
- [40] A. Katharopoulos and F. Fleuret, “Not all samples are created equal: Deep learning with importance sampling,” in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2530–2539. [Online]. Available: <http://proceedings.mlr.press/v80/katharopoulos18a.html>
- [41] J. Fernandez and D. Downey, “Sampling informative training data for RNN language models,” in *Proceedings of ACL, Student Research Workshop*, V. Shwartz, J. Tabassum, R. Voigt, W. Che, M. de Marneffe, and M. Nissim, Eds. Association for Computational Linguistics, 2018, pp. 9–13. [Online]. Available: <https://aclanthology.org/P18-3002/>
- [42] K. Vodrahalli, K. Li, and J. Malik, “Are all training examples created equal? an empirical study,” *CoRR*, vol. abs/1811.12569, 2018. [Online]. Available: <http://arxiv.org/abs/1811.12569>
- [43] O. H. Nagoor, J. Whittle, J. Deng, B. Mora, and M. W. Jones, “Lossless compression for volumetric medical images using deep neural network with local sampling,” in *IEEE International Conference on Image Processing*. IEEE, 2020, pp. 2815–2819. [Online]. Available: <https://doi.org/10.1109/ICIP40778.2020.9191031>

- [44] —, “MedZip: 3D medical images lossless compressor using recurrent neural network (LSTM),” in *International Conference on Pattern Recognition*. IEEE, 2020, pp. 2874–2881. [Online]. Available: <https://doi.org/10.1109/ICPR48806.2021.9413341>
- [45] —, “Sampling strategies for learning-based 3D medical image compression,” *Machine Learning with Applications*, p. 100273, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666827022000135>
- [46] R. Byard and J. Payne-James, *Encyclopedia of Forensic and Legal Medicine*. Elsevier Science, 2015. [Online]. Available: <https://books.google.com.sa/books?id=gticBAAAQBAJ>
- [47] K. Horvath, H. Stögner, G. Weinhandel, and A. Uhl, “Experimental study on lossless compression of biometric iris data,” in *International Symposium on Image and Signal Processing and Analysis*. IEEE, 2011, pp. 379–384. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6046636
- [48] M. Sangeetha, P. Betty, and G. N. Kumar, “A biometric iris image compression using LZW and hybrid LZW coding algorithm,” in *International Conference on Innovations in Information, Embedded and Communication Systems*, 2017, pp. 1–6.
- [49] B. Rusyn, O. Lutsyk, Y. Lysak, A. Lukenyuk, and L. Pohreliuk, “Lossless image compression in the remote sensing applications,” in *IEEE First International Conference on Data Stream Mining Processing*, 2016, pp. 195–198.
- [50] G. Schaefer, R. Starosolski, and S. Y. Zhu, “An evaluation of lossless compression algorithms for medical infrared images,” in *IEEE Engineering in Medicine and Biology Annual Conference*, 2005, pp. 1673–1676.
- [51] J. Uthayakumar and T. Vengattaraman, “Performance evaluation of lossless compression techniques: An application of satellite images,” in *International Conference on Electronics, Communication and Aerospace Technology*, 2018, pp. 750–754.
- [52] O. H. Nagoor, R. Borgo, and M. W. Jones, “Data painter: A tool for colormap interaction,” in *Computer Graphics & Visual Computing*, T. R. Wan and F. Vidal, Eds. Eurographics Association, 2017, pp. 69–76. [Online]. Available: <https://doi.org/10.2312/cgvc.20171280>
- [53] “CT scan,” https://upload.wikimedia.org/wikipedia/commons/thumb/2/27/UPMCEast_CTscan.jpg/560px-UPMCEast_CTscan.jpg, Accessed: 2022-02-08.

- [54] G. N. Hounsfield, “Computerized transverse axial scanning (tomography): Part 1. description of system,” *The British journal of radiology*, vol. 46, no. 552, pp. 1016–1022, 1973.
- [55] J. C. Masdeu and R. G. Gonzalez, *Neuroimaging, Part I*. Oxford: Elsevier, 2016. [Online]. Available: <http://ebookcentral.proquest.com/lib/swansea-ebooks/detail.action?docID=4592086>
- [56] J. D. B. O’Sullivan, J. Behnsen, T. Starborg, A. S. MacDonald, A. T. Phythian-Adams, K. J. Else, S. M. Cruickshank, and P. J. Withers, “X-ray micro-computed tomography (μ CT): an emerging opportunity in parasite imaging,” *Parasitology*, vol. 145, no. 7, p. 848–854, 2018.
- [57] W. Stiller, “Basics of iterative reconstruction methods in computed tomography: A vendor-independent overview.” *European Journal of Radiology*, vol. 109, pp. 147–154, 2018.
- [58] P. Mildenerger, M. Eichelberg, and E. Martin, “Introduction to the DICOM standard,” *European Radiology*, vol. 12, no. 4, pp. 920–927, 2002.
- [59] “DICOM,” <https://www.dicomstandard.org/>, accessed: 2021-4-26.
- [60] Siemens, “Syngo ct 2012b dicom conformance statement,” Siemens AG, Germany, Tech. Rep., 2012. [Online]. Available: www.siemens.com/healthcare
- [61] L. H. U and H. Xie, Eds., *Web and Big Data - APWeb-WAIM International Workshops: MWDA, BAH, KGMA, DMMOOC, DS, Macau*, ser. Lecture Notes in Computer Science. Springer, 2018, vol. 11268. [Online]. Available: <https://doi.org/10.1007/978-3-030-01298-4>
- [62] K. Lawonn, N. N. Smit, K. Bühler, and B. Preim, “A survey on multimodal medical data visualization,” *Comput. Graph. Forum*, vol. 37, no. 1, pp. 413–438, 2018. [Online]. Available: <https://doi.org/10.1111/cgf.13306>
- [63] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf, *Real-time volume graphics*. A K Peters, 2006. [Online]. Available: <http://downloads.akpeters.com/product.asp?ProdCode=2663>
- [64] M. R. Stytz, G. Frieder, and O. Frieder, “Three-dimensional medical imaging: Algorithms and computer systems,” *ACM Comput. Surv.*, vol. 23, no. 4, p. 421–499, dec 1991. [Online]. Available: <https://doi.org/10.1145/125137.125155>
- [65] A. Kaushik, G. Bhutani, P. Venkata, T. Dwarakanath, and A. Moiyadi, “Image based data preparation for neuronavigation,” in *International and National Conference on Machines and Mechanisms*, 2015.

- [66] J. Kniss, G. L. Kindlmann, and C. D. Hansen, “Multidimensional transfer functions for interactive volume rendering,” *IEEE Trans. Vis. Comput. Graph.*, vol. 8, no. 3, pp. 270–285, 2002. [Online]. Available: <https://doi.org/10.1109/TVCG.2002.1021579>
- [67] M. Berger, J. Li, and J. A. Levine, “A generative model for volume rendering,” *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 4, pp. 1636–1650, 2019. [Online]. Available: <https://doi.org/10.1109/TVCG.2018.2816059>
- [68] Y. Sato, C. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis, “Tissue classification based on 3D local intensity structures for volume rendering,” *IEEE Trans. Vis. Comput. Graph.*, vol. 6, no. 2, pp. 160–180, 2000. [Online]. Available: <https://doi.org/10.1109/2945.856997>
- [69] D. Shao, Y. Dai, N. Li, X. Cao, W. Zhao, L. Cheng, Z. Rong, L. Huang, Y. Wang, and J. Zhao, “Artificial intelligence in clinical research of cancers,” *Briefings in Bioinformatics*, vol. 23, no. 1, 2021. [Online]. Available: <https://doi.org/10.1093/bib/bbab523>
- [70] A. Hosny, C. Parmar, J. Quackenbush, L. H. Schwartz, and H. J. Aerts, “Artificial intelligence in radiology,” *Nature Reviews Cancer*, vol. 18, pp. 500–510, 2018.
- [71] X. Xie, J. Niu, X. Liu, Z. Chen, S. Tang, and S. Yu, “A survey on incorporating domain knowledge into deep learning for medical image analysis,” *Medical Image Analysis*, vol. 69, p. 101985, 2021.
- [72] S. J. DenOtter TD, “Hounsfield unit.” StatPearls [Internet], Florida US, Tech. Rep., Mar. 2021. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK547721/#>
- [73] D. L. Pham and P.-L. Bazin, “Chapter 12 - unsupervised tissue classification,” in *Handbook of Medical Image Processing and Analysis (Second Edition)*, second edition ed., I. N. Bankman, Ed. Burlington: Academic Press, 2009, pp. 209–222. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123739049500209>
- [74] V. Ferreira, M. Robson, T. Karamitsos, M. Bissell, D. Tyler, and S. Neubauer, “6 - magnetic resonance imaging,” in *Advanced Cardiac Imaging*, ser. Woodhead Publishing Series in Biomaterials, K. Nieman, O. Gaemperli, P. Lancellotti, and S. Plein, Eds. Woodhead Publishing, 2015, pp. 127–169. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9781782422822000068>

- [75] J. Van Stappen, “A micro-CT investigation into pore-scale CO₂ sequestration processes in fractured reservoir rocks,” Ph.D. dissertation, 12 2017.
- [76] A. A. Almashraqi, “Optimization of Radiation Dose and Image Quality for Large Field of View Cone-beam Computed Tomography: An In Vitro Study,” *World Journal of Dentistry*, vol. 11, pp. 346–354, 2020.
- [77] Y. Thakur, P. Mclaughlin, and J. R. Mayo, “Strategies for radiation dose optimization,” *Current Radiology Reports*, vol. 1, pp. 1–10, 2013.
- [78] V. Tsapaki, “Radiation dose optimization in diagnostic and interventional radiology: Current issues and future perspectives,” *Physica Medica: European Journal of Medical Physics*, vol. 79, pp. 16–21, 2020. [Online]. Available: <https://doi.org/10.1016/j.ejmp.2020.09.015>
- [79] K. Sayood, “Chapter 1 - introduction,” in *Introduction to Data Compression (Fifth Edition)*, ser. The Morgan Kaufmann Series in Multimedia Information and Systems, K. Sayood, Ed. Morgan Kaufmann, 2018, pp. 1–10. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B978012809474700001X>
- [80] K. Ramya and M.Pushpa, “A survey on lossless and lossy data compression methods,” *International Journal of Computer Science and Engineering Communications*, vol. 4, 2016.
- [81] R. M. Thanki and A. Kothari, *Data Compression and Its Application in Medical Imaging*. Cham: Springer International Publishing, 2019, pp. 1–15. [Online]. Available: https://doi.org/10.1007/978-3-030-12575-2_1
- [82] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004. [Online]. Available: <https://doi.org/10.1109/TIP.2003.819861>
- [83] A. Horé and D. Ziou, “Image quality metrics: PSNR vs. SSIM,” in *International Conference on Pattern Recognition*. IEEE Computer Society, 2010, pp. 2366–2369. [Online]. Available: <https://doi.org/10.1109/ICPR.2010.579>
- [84] S. Brunton and J. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. [Online]. Available: <https://books.google.com.sa/books?id=CYaEDwAAQBAJ>
- [85] V. Britanak, P. C. Yip, and K. Rao, “Chapter 4 - fast dct/dst algorithms,” in *Discrete Cosine and Sine Transforms*, V. Britanak, P. C. Yip, and

- K. Rao, Eds. Oxford: Academic Press, 2007, pp. 73–140. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123736246500060>
- [86] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression,” *IEEE Trans. Inf. Theory*, vol. 23, no. 3, pp. 337–343, 1977. [Online]. Available: <https://doi.org/10.1109/TIT.1977.1055714>
- [87] —, “Compression of individual sequences via variable-rate coding,” *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, 1978. [Online]. Available: <https://doi.org/10.1109/TIT.1978.1055934>
- [88] T. A. Welch, “A technique for high-performance data compression,” *Computer*, vol. 17, no. 6, pp. 8–19, 1984. [Online]. Available: <https://doi.org/10.1109/MC.1984.1659158>
- [89] G. G. Langdon, “An introduction to arithmetic coding,” *IBM J. Res. Dev.*, vol. 28, no. 2, pp. 135–149, 1984. [Online]. Available: <https://doi.org/10.1147/rd.282.0135>
- [90] J. Rissanen and G. G. Langdon, “Arithmetic coding,” *IBM J. Res. Dev.*, vol. 23, no. 2, pp. 149–162, 1979. [Online]. Available: <https://doi.org/10.1147/rd.232.0149>
- [91] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proceedings of the Institute of Radio Engineers*, vol. 40, no. 9, pp. 1098–1101, 1952.
- [92] S. W. Golomb, “Run-length encodings (corresp.),” *IEEE Trans. Inf. Theory*, vol. 12, no. 3, pp. 399–401, 1966. [Online]. Available: <https://doi.org/10.1109/TIT.1966.1053907>
- [93] G. K. Wallace, “The JPEG still picture compression standard,” *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991. [Online]. Available: <https://doi.org/10.1145/103085.103089>
- [94] D. L. Gall, “MPEG: A video compression standard for multimedia applications,” *Commun. ACM*, vol. 34, no. 4, pp. 46–58, 1991. [Online]. Available: <https://doi.org/10.1145/103085.103090>
- [95] K. Brandenburg, “MP3 and AAC explained,” in *Audio Engineering Society Conference: International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.

- [96] S. Shanmugasundaram and R. Lourdasamy, “A comparative study of text compression algorithms,” *International Journal of Wisdom Based Computing*, vol. 1, no. 3, pp. 68–76, 2011.
- [97] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948. [Online]. Available: <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>
- [98] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org/>
- [99] C. M. Bishop, *Pattern recognition and machine learning, 5th Edition*, ser. Information science and statistics. Springer, 2007. [Online]. Available: <https://www.worldcat.org/oclc/71008143>
- [100] J. G. Cleary and I. H. Witten, “Data compression using adaptive coding and partial string matching,” *IEEE Trans. Commun.*, vol. 32, no. 4, pp. 396–402, 1984. [Online]. Available: <https://doi.org/10.1109/TCOM.1984.1096090>
- [101] A. Moffat, “Implementing the PPM data compression scheme,” *IEEE Trans. Commun.*, vol. 38, no. 11, pp. 1917–1921, 1990. [Online]. Available: <https://doi.org/10.1109/26.61469>
- [102] P. G. Howard, *The Design and Analysis of Efficient Lossless Data Compression Systems*. USA: Brown University, 1993.
- [103] I. Pavlov, “7z format,” <http://www.7zip.org/7z.html>, accessed: 2021-4-26.
- [104] J. A. Storer and T. G. Szymanski, “Data compression via textual substitution,” *J. ACM*, vol. 29, no. 4, pp. 928–951, 1982. [Online]. Available: <https://doi.org/10.1145/322344.322346>
- [105] A. Moffat and A. Turpin, *Compression and Coding Algorithms*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 2012. [Online]. Available: https://books.google.com.sa/books?id=-f_SBwAAQBAJ
- [106] S. Shanmugasundaram and R. Lourdasamy, “A comparative study of text compression algorithms,” *International Journal of Wisdom Based Computing*, vol. 1, no. 3, pp. 68–76, 2011.
- [107] H. Park, G. R. Martin, and A. Bhalerao, “An affine symmetric image model and its applications,” *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1695–1705, 2010. [Online]. Available: <https://doi.org/10.1109/TIP.2010.2045692>

- [108] —, “Local affine image matching and synthesis based on structural patterns,” *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 1968–1977, 2010. [Online]. Available: <https://doi.org/10.1109/TIP.2010.2045704>
- [109] C. Westin, A. Bhalerao, R. Kikinis, and H. Knutsson, “Using local 3D structure for segmentation of bone from computer tomography images,” in *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1997, pp. 794–800. [Online]. Available: <https://doi.org/10.1109/CVPR.1997.609418>
- [110] C. Westin, S. K. Warfield, A. Bhalerao, L. Mui, J. A. Richolt, and R. Kikinis, “Tensor controlled local structure enhancement of CT images for bone segmentation,” in *International Conference of Medical Image Computing and Computer-Assisted Intervention*, ser. Lecture Notes in Computer Science, W. M. W. III, A. C. F. Colchester, and S. L. Delp, Eds., vol. 1496. Springer, 1998, pp. 1205–1212. [Online]. Available: <https://doi.org/10.1007/BFb0056310>
- [111] C. C. Reyes-Aldasoro and A. Bhalerao, “Volumetric texture segmentation by discriminant feature selection and multiresolution classification,” *IEEE Trans. Medical Imaging*, vol. 26, no. 1, pp. 1–14, 2007. [Online]. Available: <https://doi.org/10.1109/TMI.2006.884637>
- [112] L. Wang, A. Bhalerao, and R. Wilson, “Robust modelling of local image structures and its application to medical imagery,” in *International Conference on Pattern Recognition*. IEEE Computer Society, 2004, pp. 534–537. [Online]. Available: <https://doi.org/10.1109/ICPR.2004.1334584>
- [113] Q. Zhang, A. Bhalerao, C. Parsons, E. Helm, and C. Hutchinson, “Wavelet appearance pyramids for landmark detection and pathology classification: Application to lumbar spinal stenosis,” in *International Conference of Medical Image Computing and Computer-Assisted Intervention*, ser. Lecture Notes in Computer Science, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. B. Ünal, and W. Wells, Eds., vol. 9901, 2016, pp. 274–282. [Online]. Available: https://doi.org/10.1007/978-3-319-46723-8_32
- [114] J. B. J. baron Fourier, *Théorie analytique de la chaleur*. Chez Firmin Didot, père et fils, 1822.
- [115] M. Carter, “10 - photographs,” in *Designing Science Presentations*, M. Carter, Ed. San Diego: Academic Press, 2013, pp. 137–150. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123859693000106>

- [116] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org/>
- [117] C. Bishop, “Pattern recognition and machine learning (information science and statistics),” in *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 2006.
- [118] K. P. Murphy, *Machine learning - a probabilistic perspective*, ser. Adaptive computation and machine learning series. MIT Press, 2012.
- [119] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*, ser. Springer Series in Statistics. Springer, 2009. [Online]. Available: <https://doi.org/10.1007/978-0-387-84858-7>
- [120] J. Brownlee, *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Jason Brownlee, 2017. [Online]. Available: <https://books.google.com.sa/books?id=ONpdsWEACAAJ>
- [121] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [122] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model,” in *INTERSPEECH Annual Conference of the International Speech Communication Association*, T. Kobayashi, K. Hirose, and S. Nakamura, Eds. ISCA, 2010, pp. 1045–1048. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html
- [123] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>
- [124] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: <http://arxiv.org/abs/1412.3555>
- [125] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the International Conference on Machine Learning*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 807–814. [Online]. Available: <https://icml.cc/Conferences/2010/papers/432.pdf>

- [126] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *CoRR*, vol. abs/1505.00853, 2015. [Online]. Available: <http://arxiv.org/abs/1505.00853>
- [127] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *International Conference on Computational Statistics*, Y. Lechevallier and G. Saporta, Eds. Physica-Verlag, 2010, pp. 177–186. [Online]. Available: https://doi.org/10.1007/978-3-7908-2604-3_16
- [128] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [129] M. Li, T. Zhang, Y. Chen, and A. J. Smola, “Efficient mini-batch training for stochastic optimization,” in *The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, S. A. Macskassy, C. Perlich, J. Leskovec, W. Wang, and R. Ghani, Eds. ACM, 2014, pp. 661–670. [Online]. Available: <https://doi.org/10.1145/2623330.2623612>
- [130] K. Smagulova and A. P. James, “A survey on lstm memristive neural network architectures and applications,” *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019. [Online]. Available: <https://doi.org/10.1140/epjst/e2019-900046-x>
- [131] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, “Image transformer,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4055–4064.
- [132] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: <https://doi.org/10.18653/v1/n19-1423>
- [133] V. Sridhar, “A review of the effective techniques of compression in medical image processing,” *International Journal of Computer Applications*, vol. 97, no. 6, 2014.
- [134] X. Wu and N. D. Memon, “CALIC—a context based adaptive lossless image codec,” in *IEEE International Conference on Acoustics, Speech, and Signal*

- Processing Conference Proceedings*. IEEE Computer Society, 1996, pp. 1890–1893. [Online]. Available: <https://doi.org/10.1109/ICASSP.1996.544819>
- [135] I. Matsuda, H. Mori, and S. Itoh, “Lossless coding of still images using minimum-rate predictors,” in *Proceedings of the 2000 International Conference on Image Processing*. IEEE, 2000, pp. 132–135. [Online]. Available: <https://doi.org/10.1109/ICIP.2000.900912>
- [136] X. Wu and N. D. Memon, “Context-based lossless interband compression-extending CALIC,” *IEEE Trans. Image Process.*, vol. 9, no. 6, pp. 994–1001, 2000. [Online]. Available: <https://doi.org/10.1109/83.846242>
- [137] E. Magli, G. Olmo, and E. Quacchio, “Optimized onboard lossless and near-lossless compression of hyperspectral data using CALIC,” *IEEE Geosci. Remote. Sens. Lett.*, vol. 1, no. 1, pp. 21–25, 2004. [Online]. Available: <https://doi.org/10.1109/LGRS.2003.822312>
- [138] I. Schiopu and A. Munteanu, “Deep-learning-based lossless image coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1829–1842, 2020. [Online]. Available: <https://doi.org/10.1109/TCSVT.2019.2909821>
- [139] —, “A study of prediction methods based on machine learning techniques for lossless image coding,” in *IEEE International Conference on Image Processing*. IEEE, 2020, pp. 3324–3328. [Online]. Available: <https://doi.org/10.1109/ICIP40778.2020.9190696>
- [140] I. Schiopu, H. Huang, and A. Munteanu, “Cnn-based intra-prediction for lossless HEVC,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1816–1828, 2020. [Online]. Available: <https://doi.org/10.1109/TCSVT.2019.2940092>
- [141] H. Rhee, Y. I. Jang, S. Kim, and N. I. Cho, “Channel-wise progressive learning for lossless image compression,” in *IEEE International Conference on Image Processing*. IEEE, 2020, pp. 1113–1117. [Online]. Available: <https://doi.org/10.1109/ICIP40778.2020.9191322>
- [142] D. S. Taubman and M. W. Marcellin, *JPEG2000 - image compression fundamentals, standards and practice*, ser. The Kluwer international series in engineering and computer science. Kluwer, 2002, vol. 642. [Online]. Available: <https://doi.org/10.1007/978-1-4615-0799-4>
- [143] J. Townsend, T. Bird, and D. Barber, “Practical lossless compression with latent variables using bits back coding,” in *International Conference on*

- Learning Representations*. OpenReview.net, 2019, pp. –. [Online]. Available: <https://openreview.net/forum?id=ryE98iR5tm>
- [144] F. H. Kingma, P. Abbeel, and J. Ho, “Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables,” in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 3408–3417. [Online]. Available: <http://proceedings.mlr.press/v97/kingma19a.html>
- [145] M. Goyal, K. Tatwawadi, S. Chandak, and I. Ochoa, “Deepzip: Lossless data compression using recurrent neural networks,” in *Data Compression Conference*, A. Bilgin, M. W. Marcellin, J. Serra-Sagrìstà, and J. A. Storer, Eds. IEEE, 2019, p. 575. [Online]. Available: <https://doi.org/10.1109/DCC.2019.00087>
- [146] B. Knoll., “lstm-compress: data compression using LSTM,” <https://github.com/byronknoll/lstm-compress>, 2019, accessed: 15.07.2020.
- [147] Y. Hu, W. Yang, Z. Ma, and J. Liu, “Learning end-to-end lossy image compression: A benchmark,” *CoRR*, vol. abs/2002.03711, 2020. [Online]. Available: <https://arxiv.org/abs/2002.03711>
- [148] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, “Full resolution image compression with recurrent neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2017, pp. 5435–5443. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.577>
- [149] A. S. Sushmit, S. U. Zaman, A. I. Humayun, T. Hasan, and M. I. H. Bhuiyan, “X-ray image compression using convolutional recurrent neural networks,” in *IEEE EMBS International Conference on Biomedical Health Informatics*, May 2019, pp. 1–4.
- [150] J. Whittle and M. W. Jones, “A Deep Learning Approach to No-Reference Image Quality Assessment For Monte Carlo Rendered Images,” in *Computer Graphics & Visual Computing*, G. K. L. Tam and F. Vidal, Eds. Eurographics Association, 2018, pp. 23–31. [Online]. Available: <https://doi.org/10.2312/cgvc.20181204>
- [151] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2016, pp. 1646–1654. [Online]. Available: <https://doi.org/10.1109/CVPR.2016.182>

- [152] W. Lai, J. Huang, N. Ahuja, and M. Yang, “Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2017, pp. 5835–5843. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.618>
- [153] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, “Photo-realistic single image super-resolution using a generative adversarial network,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2017, pp. 105–114. [Online]. Available: <https://doi.org/10.1109/CVPR.2017.19>
- [154] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, “ESRGAN: enhanced super-resolution generative adversarial networks,” in *European Conference on Computer Vision Workshops*, ser. Lecture Notes in Computer Science, L. Leal-Taixé and S. Roth, Eds., vol. 11133. Springer, 2018, pp. 63–79. [Online]. Available: https://doi.org/10.1007/978-3-030-11021-5_5
- [155] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *Proceedings of the International Conference on Machine Learning*, ser. JMLR Workshop and Conference Proceedings, M. Balcan and K. Q. Weinberger, Eds., vol. 48. JMLR.org, 2016, pp. 1747–1756. [Online]. Available: <http://proceedings.mlr.press/v48/oord16.html>
- [156] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, “PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications,” in *International Conference on Learning Representations*. OpenReview.net, 2017, pp. –. [Online]. Available: <https://openreview.net/forum?id=BJrFC6ceg>
- [157] S. E. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. de Freitas, “Parallel multiscale autoregressive density estimation,” in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 2912–2921. [Online]. Available: <http://proceedings.mlr.press/v70/reed17a.html>
- [158] C. A. Christopoulos, A. N. Skodras, and T. Ebrahimi, “The JPEG2000 still image coding system: an overview,” *IEEE Trans. Consumer Electron.*, vol. 46, no. 4, pp. 1103–1127, 2000. [Online]. Available: <https://doi.org/10.1109/30.920468>

- [159] “HEVC Format Range Extension (RExt).” <https://hevc.hhi.fraunhofer.de/rext>, 2019, accessed: 15.07.2019.
- [160] A. J. Hussain, A. Al-Fayadh, and N. Radi, “Image compression techniques: A survey in lossless and lossy algorithms,” *Neurocomputing*, vol. 300, pp. 44–69, 2018. [Online]. Available: <https://doi.org/10.1016/j.neucom.2018.02.094>
- [161] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, “Deep learning-based video coding: A review and a case study,” *ACM Comput. Surv.*, vol. 53, no. 1, pp. 11:1–11:35, 2020. [Online]. Available: <https://doi.org/10.1145/3368405>
- [162] X. Yi, E. Walia, and P. S. Babyn, “Generative adversarial network in medical imaging: A review,” *Medical Image Anal.*, vol. 58, 2019. [Online]. Available: <https://doi.org/10.1016/j.media.2019.101552>
- [163] I. Schiopu and A. Munteanu, “Macro-Pixel Prediction Based on Convolutional Neural Networks for Lossless Compression of Light Field Images,” in *IEEE International Conference on Image Processing*. IEEE, 2018, pp. 445–449. [Online]. Available: <https://doi.org/10.1109/ICIP.2018.8451731>
- [164] Z. Jiang, W. D. Pan, and H. Shen, “Spatially and spectrally concatenated neural networks for efficient lossless compression of hyperspectral imagery,” *Journal of Imaging*, vol. 6, no. 6, 2020. [Online]. Available: <https://www.mdpi.com/2313-433X/6/6/38>
- [165] I. Schiopu, Y. Liu, and A. Munteanu, “CNN-based Prediction for Lossless Coding of Photographic Images,” in *Picture Coding Symposium*. IEEE, 2018, pp. 16–20. [Online]. Available: <https://doi.org/10.1109/PCS.2018.8456311>
- [166] D. Xue, H. Ma, L. Li, D. Liu, and Z. Xiong, “iWave3D: End-to-end Brain Image Compression with Trainable 3-D Wavelet Transform,” in *International Conference on Visual Communications and Image Processing*. IEEE, 2021, pp. 1–5. [Online]. Available: <https://doi.org/10.1109/VCIP53242.2021.9675359>
- [167] L. F. R. Lucas, N. M. M. Rodrigues, L. A. da Silva Cruz, and S. M. M. de Faria, “Lossless compression of medical images using 3-D predictors,” *IEEE Trans. Medical Imaging*, vol. 36, no. 11, pp. 2250–2260, 2017. [Online]. Available: <https://doi.org/10.1109/TMI.2017.2714640>
- [168] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” in *International Conference on Learning Representations*. OpenReview.net, 2017. [Online]. Available: <https://openreview.net/forum?id=rJiNwv9gg>

- [169] S. Santurkar, D. M. Budden, and N. Shavit, “Generative compression,” in *Picture Coding Symposium*. IEEE, 2018, pp. 258–262. [Online]. Available: <https://doi.org/10.1109/PCS.2018.8456298>
- [170] Y. Liu, Y. Wang, L. Deng, F. Wang, F. Liu, Y. Lu, and S. Li, “A novel in situ compression method for CFD data based on generative adversarial network,” *J. Vis.*, vol. 22, no. 1, pp. 95–108, 2019. [Online]. Available: <https://doi.org/10.1007/s12650-018-0519-x>
- [171] M. U. Ayoobkhan, E. Chikkannan, and K. Ramakrishnan, “Feed-forward neural network-based predictive image coding for medical image compression,” *The Arabian Journal for Science and Engineering*, vol. 43, no. 8, pp. 4239–4247, Aug 2018. [Online]. Available: <https://doi.org/10.1007/s13369-017-2837-z>
- [172] D. Flynn, D. Marpe, M. Naccari, T. Nguyen, C. Rosewarne, K. Sharman, J. Sole, and J. Xu, “Overview of the range extensions for the HEVC standard: Tools, profiles, and performance,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 4–19, 2016. [Online]. Available: <https://doi.org/10.1109/TCSVT.2015.2478707>
- [173] “DCMTK is a collection of libraries, which includes DCMCJPLS tool for encoding DICOM file to JPEG-LS transfer syntax,” <https://support.dcmtk.org/docs/dcmcjpls.html>, 2020, accessed: 15.07.2020.
- [174] “Openjpeg an open-source jpeg 2000 codec written in c.” <https://www.openjpeg.org/>, 2019, accessed: 15.07.2019.
- [175] “7-zip is a file archiver with a high compression ratio.” <https://www.7-zip.org/>, 2019, accessed: 15.06.2019.
- [176] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep Laplacian pyramid networks for fast and accurate super-resolution,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 624–632.
- [177] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [178] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, “A high-bias, low-variance introduction to machine learning for physicists,” *Physics Reports*, vol. 810, pp. 1–124, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0370157319300766>
- [179] X. Peng, L. Li, and F. Wang, “Accelerating minibatch stochastic gradient descent using typicality sampling,” *IEEE Trans. Neural Networks*

- Learn. Syst.*, vol. 31, no. 11, pp. 4649–4659, 2020. [Online]. Available: <https://doi.org/10.1109/TNNLS.2019.2957003>
- [180] G. Ioannou, T. Tagaris, and A. Stafylopatis, “Improving the convergence speed of deep neural networks with biased sampling,” in *The International Conference on Advances in Artificial Intelligence*. ACM, 2019, pp. 35–41. [Online]. Available: <https://doi.org/10.1145/3369114.3369116>
- [181] D. Csiba and P. Richtárik, “Importance sampling for minibatches,” *The Journal of Machine Learning Research*, vol. 19, 2018. [Online]. Available: <http://jmlr.org/papers/v19/16-241.html>
- [182] Z. E. Mariet, J. Robinson, J. A. Smith, S. Sra, and S. Jegelka, “Optimal batch variance with second-order marginals,” 2020.
- [183] A. El Korchi and Y. Ghanou, “Unrestricted random sampling of data batch to improve the efficiency of neural networks,” in *Proceedings of the New Challenges in Data Sciences: Acts of the Second Conference of the Moroccan Classification Society*. Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3314074.3314102>
- [184] B. Hadjadji and Y. Chibani, “Optimized selection of training samples for one-class neural network classifier,” in *International Joint Conference on Neural Networks*. IEEE, 2014, pp. 345–349. [Online]. Available: <https://doi.org/10.1109/IJCNN.2014.6889429>
- [185] A. Stewart, “Basic statistics and epidemiology: A practical guide, fourth edition,” 2016.
- [186] S. T. Acton, “Chapter 20 - diffusion partial differential equations for edge detection,” in *The Essential Guide to Image Processing*, A. Bovik, Ed. Academic Press, 2009, pp. 525–552. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123744579000202>
- [187] K. H. Hohne and R. Bernstein, “Shading 3D-Images from CT Using Gray-Level Gradients,” *IEEE Transactions on Medical Imaging*, vol. 5, no. 1, pp. 45–47, 1986.
- [188] B. Preim, C. P. Botha, and D. Bartz, *Visual Computing for Medicine: Theory, Algorithms, and Applications*. Elsevier Science & Technology, 2013. [Online]. Available: <http://ebookcentral.proquest.com/lib/swansea-ebooks/detail.action?docID=1561058>

APPENDICES

A Papers Submitted and Under Preparation

- O. H. Nagoor, R. Borgo, and M. W. Jones, “Data Painter: A Tool for Colormap Interaction”, *Submitted to Computer Graphics & Visual Computing*, 2017, **Awarded The Best Full Paper Award**, [52].
- O. H. Nagoor, J. Whittle, J. Deng, B. Mora, M. W. Jones, “Lossless Compression for Volumetric Medical Images Using Deep Neural Network With Local Sampling”, *Submitted to The IEEE International Conference on Image Processing*, 2020, **Awarded ICIP 2020 Top Viewed Q&A Paper Award (2nd place)**, [43].
- O. H. Nagoor, J. Whittle, J. Deng, B. Mora and M. W. Jones, “MedZip: 3D Medical Images Lossless Compressor Using Recurrent Neural Network (LSTM)”, *Submitted to The 25th International Conference on Pattern Recognition*, Jan. 2020, [44].
- Omniah H. Nagoor, Joss Whittle, Jingjing Deng, Ben Mora, and Mark W. Jones, “Sampling Strategies for Learning-based 3D Medical Image Compression”, *Accepted for Elsevier Machine Learning with Applications (MWLA) Journal*, Feb. 2022, [45].

B Appendix - Additional Experimental Figures

B.1 Chapter 4 - Additional Experimental Figures

This section introduces another version of Fig. 4.5 while ordering volumes from test set 1 based on their pixel spacing values to highlight any interesting impact.

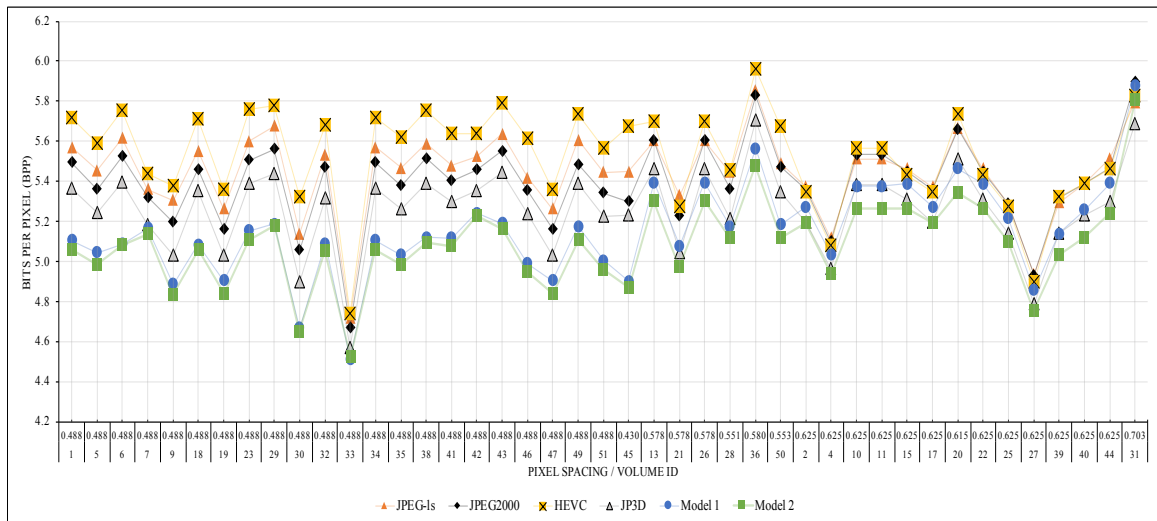


Figure B.1: Comparing the compression ratio in bpp for the proposed MLP models with the state-of-the-art lossless compression methods over 16-bits volumes on test set 1 ordered by the pixel spacing (see Fig. 4.5 for another version of this figure).

B.2 Chapter 6 - Additional Experimental Figures

This section introduces laid out of some figures (i.e. 6.7, 6.9, and 6.10) from chapter 6 in another manners to put the overall receptive density next to each other for comparison including input options with similar dimensions (i.e. 3D) and block sizes between different input shapes pyramids and cubes. For instance, considering 3D-pyramid ($7 \times 7, 5 \times 5, 3 \times 3, 1$) has much overlap with 3D-cube ($7 \times 7 \times 7$), such comparison intends to highlight the effect of dropping those other voxels.

		3D-Cube (5x5x5) - N=62	3D-Pyramid (5x5,3x3,1) - N=22	3D-Cube (7x7x7) - N=171	3D-Pyramid (7x7,5x5,3x3,1) - N=55	3D-Cube (9x9x9) - N=364	3D-Pyramid (9x9,5x5,1) - N=68	3D-Cube (11x11x11) - N=118	3D-Pyramid (13x13,9x9,5x5,1) - N=175	
Fold 1	1	4.681	4.684	4.52	4.537	4.438	4.407	4.406	4.503	4.368
	2	4.623	4.65	4.475	4.511	4.401	4.398	4.389	4.498	4.331
	3	4.72	4.723	4.558	4.57	4.496	4.446	4.439	4.53	4.418
Fold 2	4	4.742	4.739	4.64	4.636	4.681	4.61	4.616	4.766	4.577
	5	4.456	4.479	4.322	4.339	4.322	4.225	4.233	4.437	4.175
	6	4.707	4.68	4.531	4.525	4.51	4.409	4.411	4.599	4.357
Fold 3	7	4.474	4.489	4.326	4.372	4.267	4.25	4.251	4.34	4.224
	8	4.715	4.765	4.543	4.556	4.47	4.417	4.402	4.522	4.392
	9	4.806	4.829	4.649	4.653	4.595	4.517	4.522	4.622	4.488
Fold 4	10	4.19	4.2	4.059	4.042	3.978	3.939	3.936	4.072	3.913
	11	4.651	4.665	4.502	4.495	4.422	4.36	4.376	4.553	4.343
	12	4.132	4.15	4.058	4.057	4.019	4.011	4.021	4.151	4.013
	avg.	4.575	4.588	4.432	4.441	4.383	4.332	4.333	4.466	4.3
	std.	0.219	0.218	0.201	0.204	0.211	0.196	0.195	0.196	0.19




Figure B.2: Another version of Benchmark 1 results compares the overall receptive density in 3D input options next to each other from Fig. 6.7 in section 6.4.3. 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).

		3D-Cube (5x5x5) - N=59	3D-Pyramid (5x5,3x3,1) - N=19	3D-Cube (7x7x7) - N=167	3D-Pyramid (7x7,5x5,3x3,1) - N=51	3D-Cube (9x9x9) - N=359	3D-Pyramid (9x9,5x5,1) - N=61	3D-Cube (11x11x11) - N=659	3D-Pyramid (13x13,9x9,5x5,1) - N=169	
Fold 1	1	4.968	4.859	4.769	4.776	4.696	4.64	4.652	4.749	4.634
	2	4.871	4.777	4.69	4.708	4.636	4.582	4.612	4.691	4.573
	3	5.041	4.925	4.842	4.826	4.78	4.682	4.698	4.802	4.68
Fold 2	4	4.902	4.873	4.769	4.799	4.803	4.793	4.801	4.752	4.768
	5	4.684	4.621	4.51	4.527	4.496	4.437	4.453	4.421	4.404
	6	4.969	4.858	4.765	4.768	4.728	4.654	4.676	4.632	4.597
Fold 3	7	4.703	4.635	4.526	4.534	4.448	4.424	4.487	4.443	4.406
	8	4.979	4.847	4.778	4.777	4.684	4.616	4.691	4.621	4.594
	9	5.137	5.012	4.931	4.894	4.832	4.758	4.785	4.77	4.731
Fold 4	10	4.384	4.324	4.19	4.178	4.131	4.109	4.131	4.114	4.051
	11	4.898	4.792	4.712	4.684	4.656	4.581	4.625	4.62	4.541
	12	4.356	4.326	4.224	4.21	4.192	4.181	4.18	4.174	4.148
avg	4.824	4.738	4.642	4.64	4.59	4.538	4.566	4.566	4.511	
std	0.247	0.221	0.235	0.235	0.23	0.214	0.217	0.23	0.222	



Figure B.3: Another version of Benchmark 2 results compares the overall receptive density in 3D input options next to each other using models trained on **reduced** neighbourhood sequences from Fig. 6.9 in section 6.4.4. 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).

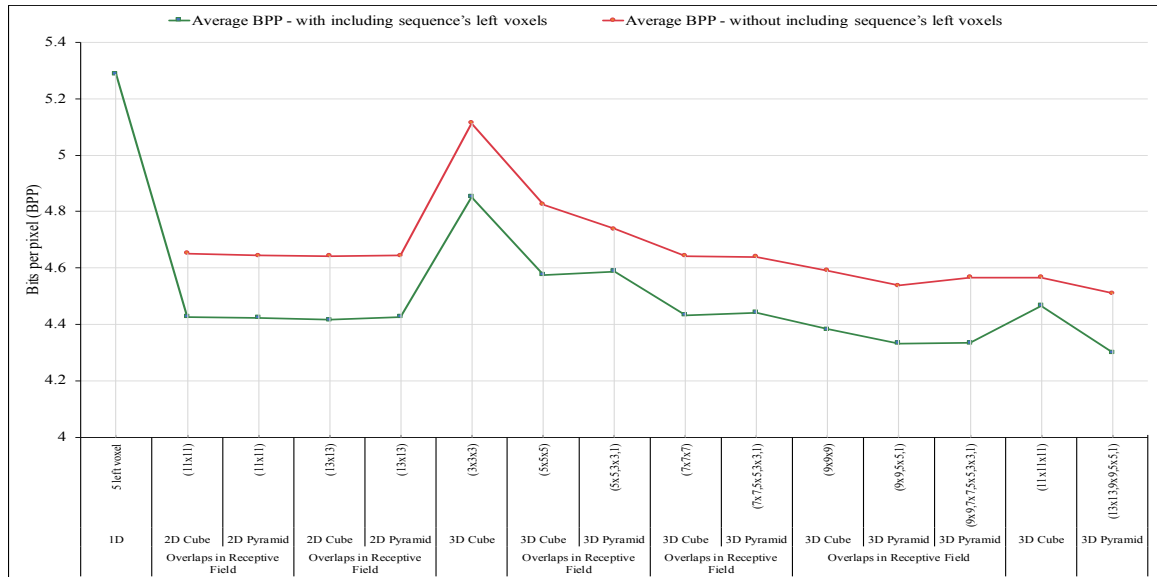
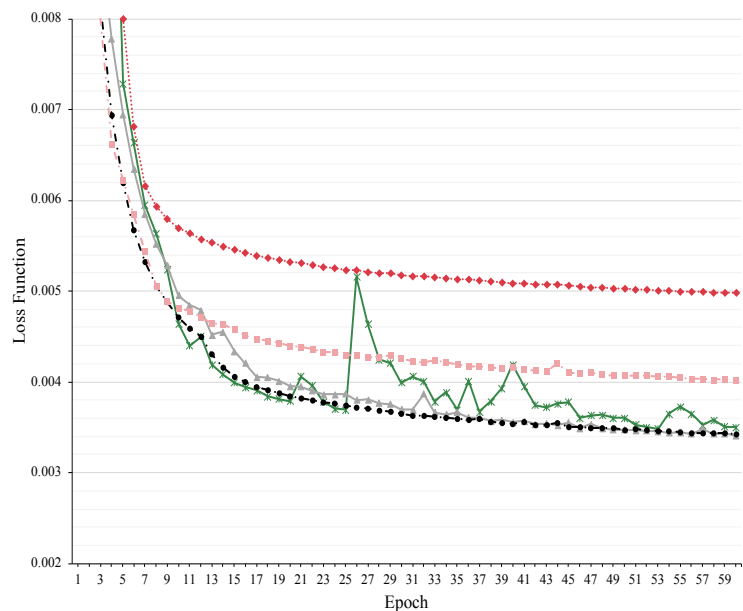
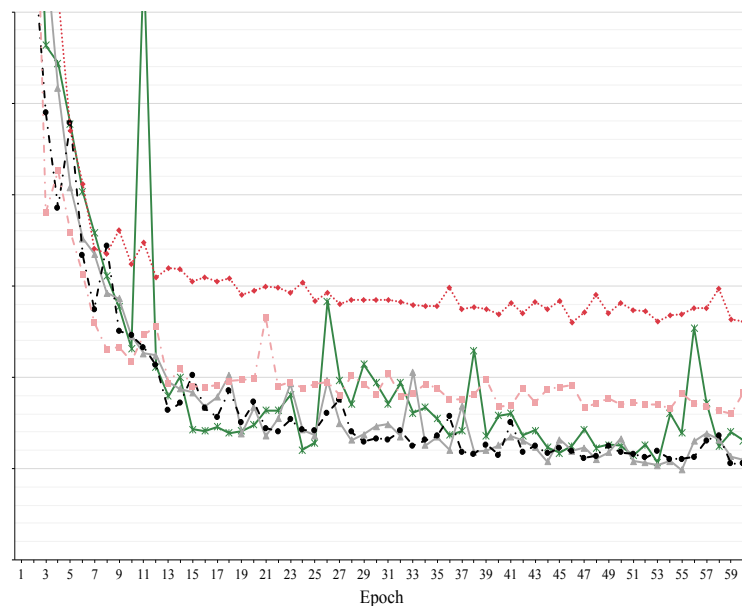


Figure B.4: An additional version of Fig. 6.10 reorders the sequence options to show the overlaps in the receptive field next to each other, which may give further insight into their contributions (see figure 6.10 and section 6.4.4 for further details).



- x— 3D Pyramid Neighboring (13x13,9x9,5x5,1) - Train
- ▲— 3D Pyramid Neighboring (9x9,7x7,5x5,3x3,1) - Train
- 3D Pyramid Neighboring (9x9,5x5,1) - Train
- 3D Pyramid Neighboring (7x7,5x5,3x3,1) - Train
- ◆— 3D Pyramid Neighboring (5x5,3x3,1) - Train

(a) Training Loss



- x— 3D Pyramid Neighboring (13x13,9x9,5x5,1) - Eval
- ▲— 3D Pyramid Neighboring (9x9,7x7,5x5,3x3,1) - Eval
- 3D Pyramid Neighboring (9x9,5x5,1) - Eval
- 3D Pyramid Neighboring (7x7,5x5,3x3,1) - Eval
- ◆— 3D Pyramid Neighboring (5x5,3x3,1) - Eval

(b) Evaluation Loss

Figure B.5: An illustration that compares model loss function plots (i.e. training loss and evaluation loss) with different pyramid input vectors (including left voxels from Benchmark1) over 60 epochs. (Another version that illustrates (bpp) variations during models' training steps is presented in Fig. 6.13.).

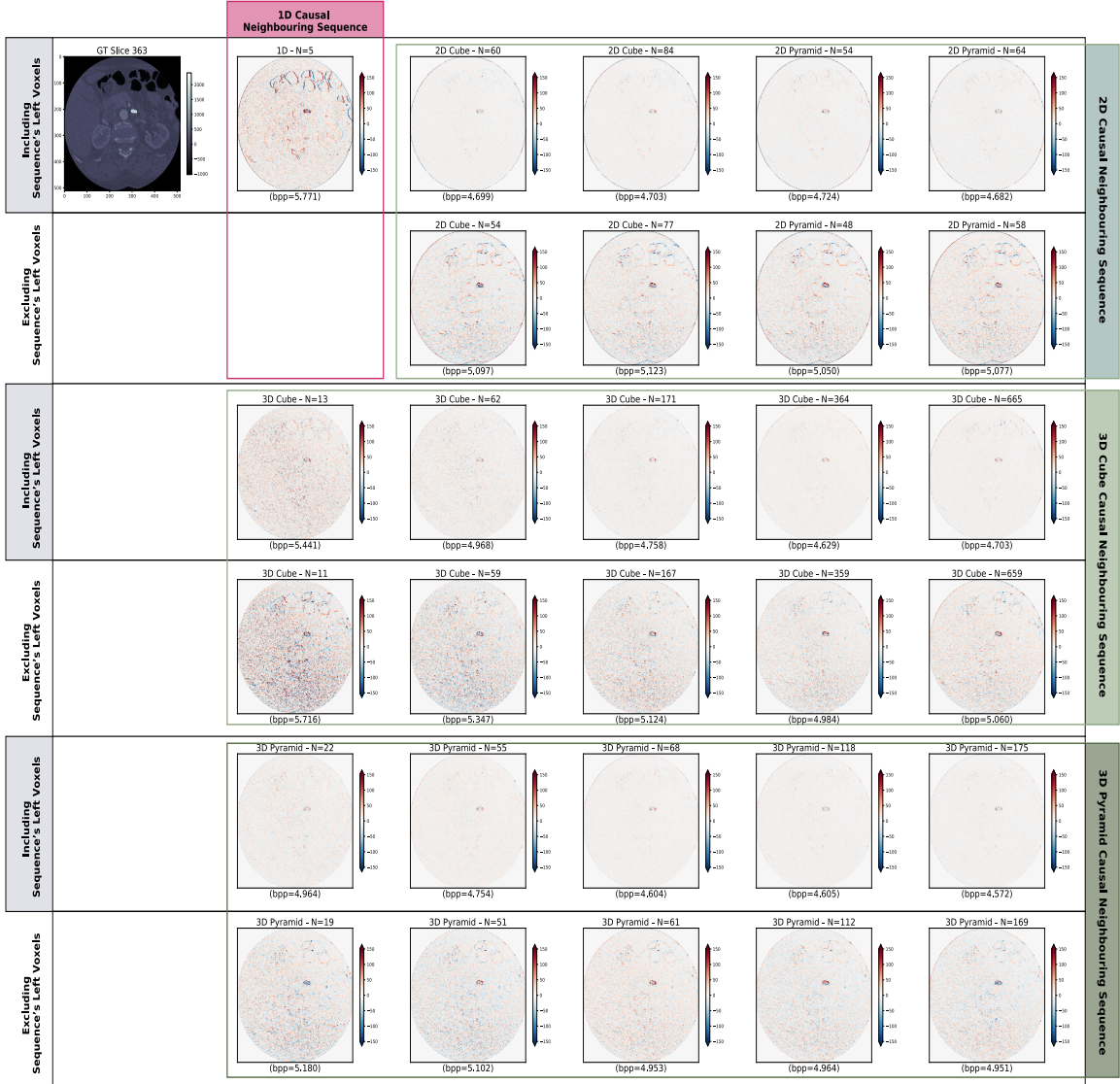


Figure B.6: Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 1. (A different version of this figure that only focuses on a specific region within the residual slices plot is presented in Fig 6.17).

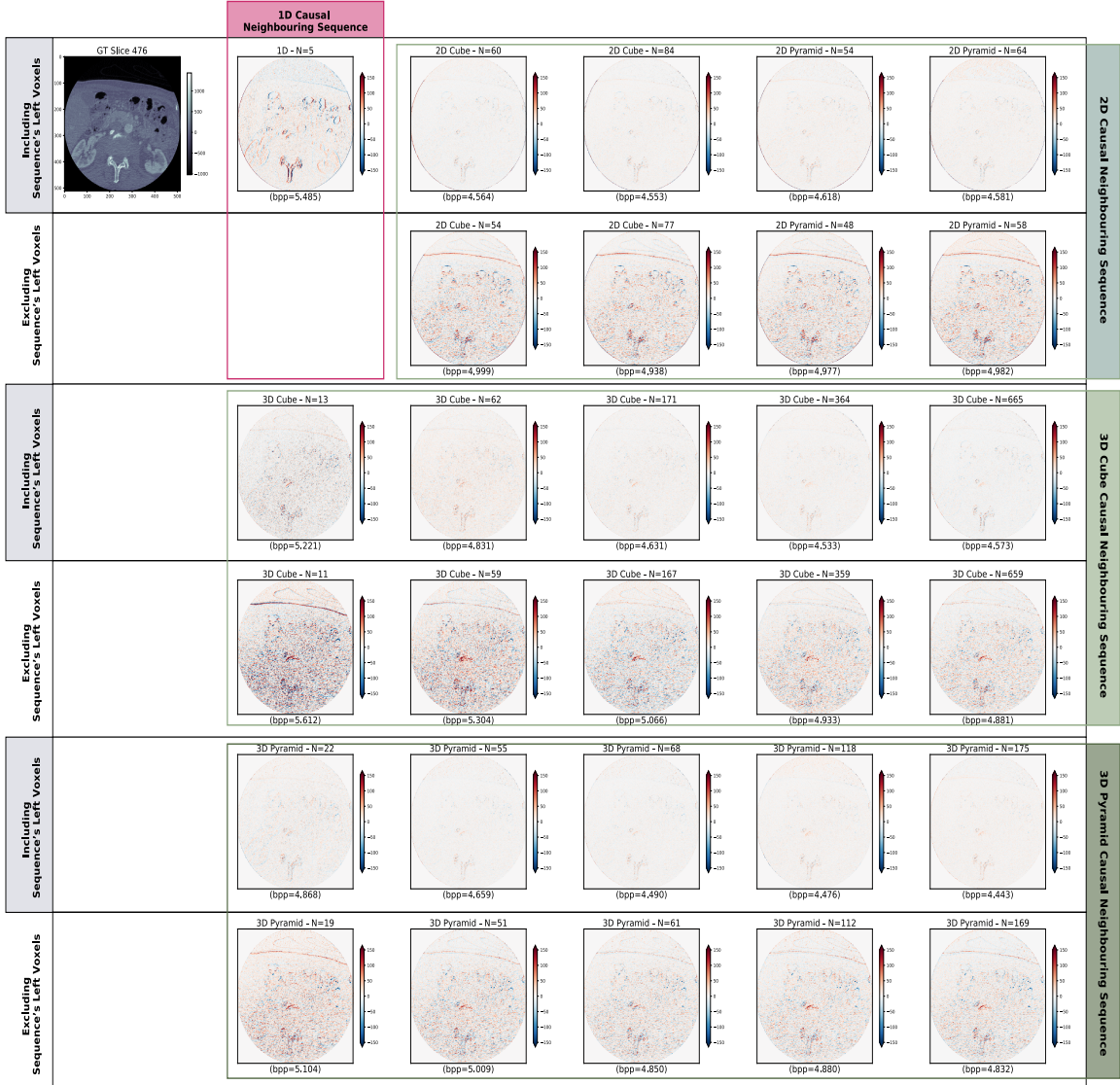


Figure B.7: Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 7. (A different version of this figure that only focuses on a specific region within the residual slices plot is presented in Fig 6.18).

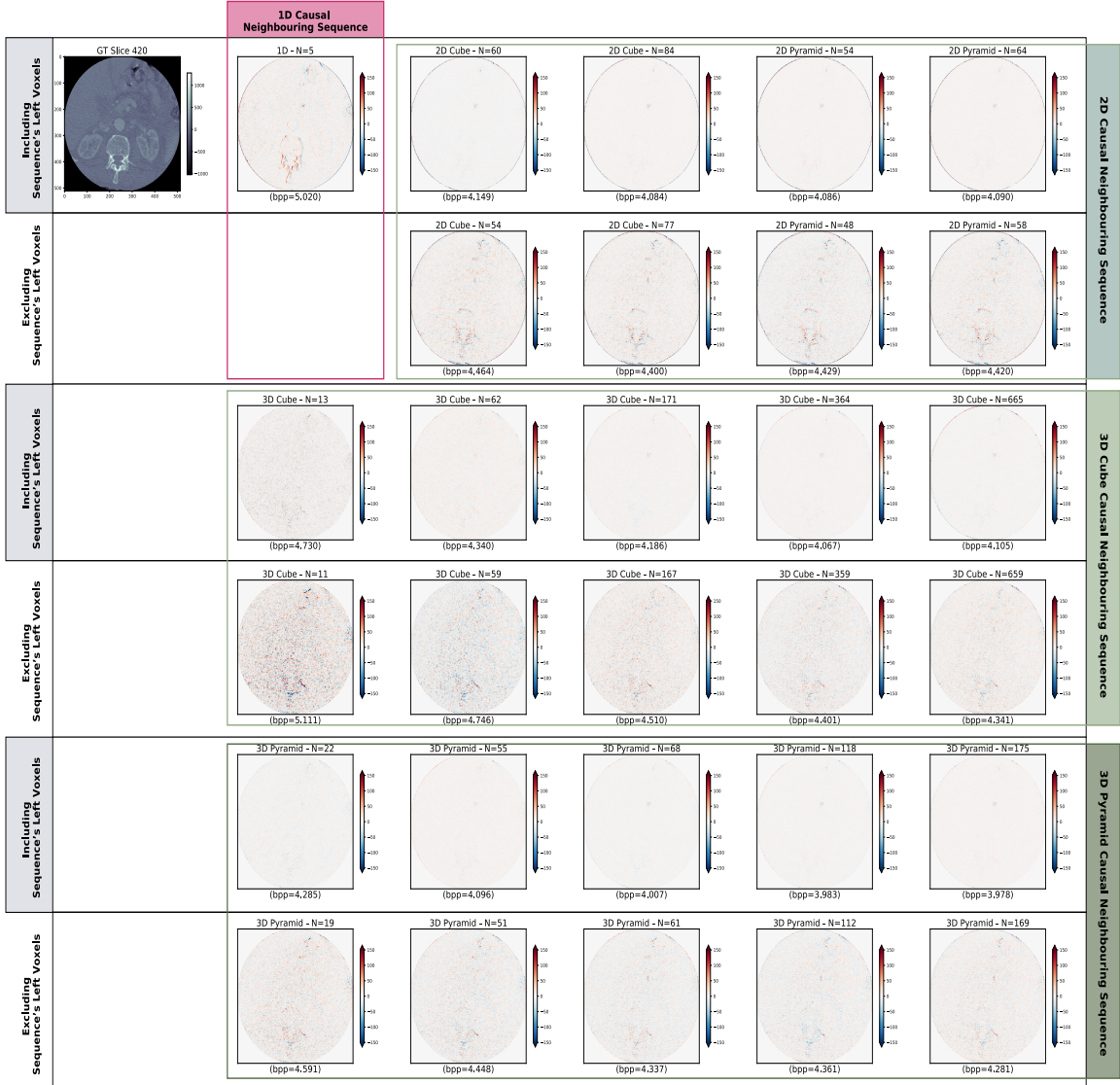


Figure B.8: Comparing the residual slices plot among the different sampling schemes for a middle slice extracted from volume 10. (A different version of this figure that only focuses on a specific region within the residual slices plot is presented in Fig 6.19).