TU WIEN Informatics

Informed Patch Sampling for 3D Medical Image Reconstruction

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Bc. Ondřej Peterka
Matrikelnummer 12229251

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dr. Renata Georgia Raidou
Mitwirkung: Gaia Romana De Paolis, MSc
Dipl.-Math. Dr. Katja Bühler

Wien, 23. März 2026

Ondřej Peterka | Renata Georgia Raidou

TU WIEN Informatics

Informed Patch Sampling for 3D Medical Image Reconstruction

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Bc. Ondřej Peterka
Registration Number 12229251

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dr. Renata Georgia Raidou
Assistance: Gaia Romana De Paolis, MSc
Dipl.-Math. Dr. Katja Bühler

Vienna, March 23, 2026

Ondřej Peterka	Renata Georgia Raidou

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Bc. Ondřej Peterka

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 23. März 2026

Ondřej Peterka

Acknowledgements

First of all, my greatest thanks belong to my biggest supporters, who have been there for me when times were tough. Without my family, I am not sure I would have ever finished or endured these stressful times. I owe them a lot—for their continuous support throughout my studies and, especially, during this thesis. Mom, Dad, David, I am glad you were there for me. I will return the favor, and that is a promise!

My second special thanks belong to my supervisor, Renata Raidou, for her helpful comments and for her assistance during my studies at TU. I do not think I have ever met a more inspiring professor—always there for the students, always fair, and always willing to help. You were the one who took good care of me when I first came to TU Wien in 2021. You were the one who inspired me to transfer to TU for my master's degree, and now you are the one helping me wrap it up and go do things in the big world. Without you, my life would not be as exciting as it is now, and for that I will always be grateful!

Lastly, I would like to thank Katja Bühler and her team at VRVis, especially Gaia and Dimitrios. I know this path was very long, but you were there for me the whole time, doing your best to make it smoother and more pleasant. Your insights and comments made it possible for me to create this thesis—a work that I am sincerely proud of—and for that, you have my thanks.

This work was enabled by the Competence Centre VRVis. The VRVis GmbH is funded by BMIMI, BMWET, Tyrol, Vorarlberg and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (911654) which is managed by FFG.

Kurzfassung

Die effiziente Analyse und Verarbeitung von 3D-Volumina ist in der klinischen Praxis unerlässlich. Eine häufige Aufgabe in der medizinischen 3D-Bildverarbeitung ist die Objektsegmentierung, bei der relevante Objekte abgegrenzt werden. Die Segmentierung großer Volumenscans erfordert jedoch erheblichen Speicherplatz und Rechenleistung, wodurch die gesamte Segmentierung speicher- und rechenintensiv ist. Ein möglicher Lösungsansatz zur Reduzierung dieser Kosten besteht darin, nur eine Teilmenge des Eingangsvolumens auszuwählen, diese zu segmentieren und die Werte der verbleibenden Regionen durch Rekonstruktion aus der segmentierten Teilmenge zu schätzen. Unsere Hypothese ist, dass die Wahl der Teilmenge die Rekonstruktionsleistung beeinflusst, wobei einige Regionen des Eingangsvolumens informativer für die Rekonstruktion sind als andere. Um diese Hypothese zu testen, schlagen wir ein neuronales Netzwerk vor, das in der Lage ist, die Teilmengen zu identifizieren, die am meisten zu einer genauen Rekonstruktion beitragen. Um den Prozess zu vereinfachen und uns auf die Kernaufgabe zu konzentrieren, gehen wir davon aus, dass binäre Segmentierungen der relevanten Objekte vorliegen, und wählen die Teilmengen direkt daraus aus, wobei die ursprünglichen binären Segmentierungen anschließend rekonstruiert werden.

Wir bauen unser neuronales Netzwerk auf einem bestehenden, auf Punktwolken basierenden Netzwerk auf, das lernt, repräsentative Punkte auszuwählen, und integrieren es in eine neuartige End-to-End-Pipeline zur Rekonstruktion vollständiger Volumina aus begrenzten Eingangsdaten. Hierzu adaptieren wir die ursprüngliche Verlustfunktion für den Einsatz auf Voxelgitterdaten und führen Konvertierungs- sowie Extraktionsmechanismen ein, die einen nahtlosen Übergang zwischen Voxelgitter- und Punktwolkenrepräsentationen gewährleisten. Unsere vorgeschlagene Verarbeitungspipeline transformiert das Eingabevolumen zunächst von seiner intrinsischen Voxelgitter-Repräsentation in eine Punktwolke, um eine effiziente geometrische Verarbeitung zu ermöglichen. Anschließend verarbeitet eine neuronale Netzwerkarchitektur diese Punktwolke und sagt eine Menge von Kandidatenzentren für volumetrische Patches voraus. Diese vorhergesagten Patch-Zentren werden anschließend genutzt, um die resultierenden Patches zu extrahieren, welche dem nachgelagerten Rekonstruktionsnetzwerk zugeführt werden.

Wir evaluieren unsere Verarbeitungspipeline anhand zweier medizinischer volumetrischer Datensätze mit vorhandenen Region-of-Interest-Annotierungen und variierender geometrischer Komplexität. Unsere Experimente zeigen, dass der vorgeschlagene trainierte

Sampler informationsreiche Regionen identifiziert, die die Rekonstruktionsleistung unterstützen, insbesondere im Fall von komplexen Formen und eingeschränktem lokalem Kontext. Darüber hinaus zeigen wir den Einfluss der Qualität des Rekonstruktionsnetzwerks über verschiedene Eingabekonfigurationen hinweg, wobei wir Patch-Größe und Patch-Anzahl variieren, und demonstrieren, dass unser Ansatz besonders dann wirksam ist, wenn die Rekonstruktionsleistung gering ist oder die Eingangsform eine komplexe Geometrie aufweist. Abschließend analysieren wir den Rechen- respektive Speicherbedarf der vorgeschlagenen Pipeline und belegen, dass der zusätzliche Overhead unter 1 GB Arbeitsspeicher und 0,5 s zusätzlicher Rechenzeit bleibt, womit die Methode für den Einsatz in ressourcenschwachen Umgebungen geeignet ist.

Abstract

Efficient analysis and processing of 3D volumes are crucial in clinical practice. A common task in 3D medical image processing is object segmentation, where objects of interest are delineated. However, segmenting large volumetric scans requires substantial memory and computational power, making end-to-end segmentation both memory- and computationally intensive. A potential solution to reduce these costs is to select only a subset of the input, segment this subset, and estimate the values of the remaining regions by reconstructing them from the segmented subset. Our hypothesis is that the choice of subset influences reconstruction performance, with some regions of the input volume being more informative for reconstruction than others. To test this hypothesis, we propose a neural network capable of identifying subsets that contribute most to accurate reconstruction. To simplify the process and focus on the core task, we assume that binary segmentations of the objects of interest are provided and select subsets directly from them, reconstructing the original binary segmentations afterwards.

We build our neural network upon an existing point cloud-based network that learns to select representative points, and integrate it into a novel end-to-end pipeline for reconstructing full volumes from limited input data. We modify the original point cloud–based loss function to operate on voxel grid data and introduce conversion and extraction mechanisms that enable transitions between voxel grid and point cloud representations. Our proposed pipeline first converts the input voxel grid into a point cloud representation to enable efficient geometric processing. A neural network architecture then processes the point cloud and predicts a set of candidate centers for volumetric patches. These predicted centers are subsequently used to extract the output patch set, which is then fed to the downstream reconstruction network.

We evaluate our pipeline on two datasets of medical shape segmentations with varying geometrical complexity. Our experiments show that the proposed learned sampler identifies informative regions, which support reconstruction performance, especially for complex shapes and limited spatial context. We further evaluate the effect of reconstruction network quality across different input configurations, varying patch size and number of patches, and show that our approach is effective when reconstruction accuracy is poor or when the input shape has complex geometry. Finally, we analyze the computations and memory demands of the proposed pipeline, showing that the additional

overhead remains under 1 GB of memory and 0.5 s of extra computation, making the method suitable for deployment in resource-limited environments.

Contents

CHAPTER 1

Introduction

3D medical image analysis plays an important role in numerous clinical applications, including disease diagnosis, abnormality detection, and treatment monitoring [MYK+25]. However, the large size and complexity of these datasets pose significant challenges for storage and processing. Efficient strategies for handling volumetric data enable scalable and practical applications in medical imaging [BTKA18]. A common task in 3D medical image analysis is object segmentation, which assigns a label of 1 to voxels corresponding to the object of interest and 0 elsewhere [ZRSTL18, TKMSH23, MYK+25]. However, state-of-the-art end-to-end segmentation models for 3D medical image processing require substantial computational resources and memory, which scale with the resolution of the volumetric data. To put this into perspective, Chen et al. [CML+24] report that current end-to-end segmentation networks require between 3.81 and 13.91 GB of memory to process two voxel grids of size $96 \times 96 \times 96$ voxels. As a result, these high resource demands limit the scalability and deployment in clinical settings [DGBCNAVA17].

In this work, we aim to tackle the high computational and memory demands of end-to-end segmentation of 3D medical images. One approach to reducing these costs is to select only a subset of the input volume, perform segmentation on this subset, and infer the remaining voxel values through reconstruction [Ho,21]. Motivated by this concept, we investigate which regions of 3D medical images are most informative, so that the full-resolution segmentation can be reconstructed with the highest possible accuracy. To isolate the core problem of the subsets selection, we simplify the problem setting and assume a binary segmentation with the object of interest delineated is given. The goal is then to select subsets from input segmentations such that, when used as input to a reconstruction model, they result in the most accurate reconstruction of the original segmentation. To achieve this, we adapt a state-of-the-art learned subset selection strategy, which identifies the most informative regions that contribute to accurate reconstruction of the input segmentation.

In Section 1.1, we present the motivation for this work by discussing the challenges in processing large volumetric medical images and proposing a more efficient alternative based on selective processing and reconstruction. Section 1.2 outlines the aim of the work and formulates the research questions addressed in this thesis. In Section 1.3, we summarize the main contributions of the proposed approach. Finally, Section 1.4 provides an overview of the structure of the thesis.

1.1 Motivation

In the medical domain, 3D data are essential for providing a comprehensive representation of anatomical structures. Modalities such as CT and MRI generate volumetric scans that allow clinicians to examine organs, tissues, and pathologies in detail, capturing precise spatial relationships that are lost in 2D representations [RJC+12]. This enables more accurate diagnosis, precise localization of abnormalities, assessment of disease progression, and improved doctor-patient communication [ZEP10].

Volumetric scans are typically stored in voxel-grids, which represent 3D space as a regular grid of cubic volume elements (voxels) [Wan24]. Such a data structure suffers from high memory requirements, which grow cubically with the size of the data [LTLH19]. Processing of volumetric data in a per-voxel manner is time-consuming and therefore often does not allow for real-time evaluation or requires significant computational power [PHH21, WXK+23]. In the medical domain, these challenges are even more prominent, as clinical environments often rely on standard workstations without access to powerful GPUs [DGBCNAVA17]. As a result, processing large volumetric scans can be slow, making tasks like segmentation or registration impractical in time-sensitive settings.

A potential solution to large-volume processing is to reduce the input size to the downstream task function [LMU21]. One way to achieve this is by selecting only a subset of the input volume, such as slices or local sub-volumes, for further processing [NMW+19, DLA19, LMA20]. This reduces the amount of data that must be stored and processed, lowering memory requirements and computational cost. Among the research community, the process of selecting a subset of input data is referred to as *sampling*, and the selected sub-volumes are referred to as *patches* or *samples* [DLA19, LMA20]. We further refer to a collection of patches sampled from a single input volume as *sparse measurements* or *sparse samples*.

Early approaches to sampling volumetric data have largely relied on heuristics, such as Random Sampling, which selects patches randomly from the input volume, or Farthest Point Sampling (FPS), which iteratively chooses the point farthest from the current set of sampled points until the desired number of samples is reached [KCDS11, VCA+17]. Such sampling heuristics follow a predefined algorithm, which is not optimized for any task or a given collection of structures. As a result, they may fail to capture important patterns in the data, leading to suboptimal performance.

Recently, learning-based sampling approaches emerged as a promising alternative, enabling

sampling strategies to be jointly optimized with a downstream task [DLA19, LMA20, YSA23, WZPB23, LLG+25]. For example, Lang et al. [LMA20] propose a point cloud-based sampling architecture, which generates a smaller (simplified) point cloud, optimized for a downstream task. However, the performance of learned approaches depends on the availability of representative datasets, and training often requires significant computational effort [TTM+22]. Moreover, designing learned samplers for volumetric data remains non-trivial due to the high dimensionality of the input, memory requirements, and the limited literature defining suitable criteria for selecting the most informative patches in the context of volumetric sampling [SEJ18]. Addressing these challenges would allow the learned sampler to be integrated into a processing pipeline in which, instead of applying the routine to the full input volume, only the most representative sub-volumes are processed.

This thesis investigates whether conventional downsampling approaches can be replaced with a more strategic selection of input patches. In particular, we analyze the case where reconstruction is used as the downstream task, allowing us to recover the full-resolution voxel grid by estimating missing values. In recent years, deep learning–based methods have become widely used for reconstruction [AÖ17, JMFU17, PFS+19, WLX+21, ALL+22, ALL+24, CKZ+24], demonstrating that high-resolution volumes can be reconstructed from sparse or incomplete data. To isolate the core problem of sampling, we simplify the problem setting and assume a binary segmentation of the 3D medical shape is provided. In this work, we design a learned volumetric sampling method capable of identifying the most informative patches and integrate it in an end-to-end pipeline, which takes as input a binary segmentation of a 3D medical image, selects the most representative patches, and reconstructs the original full-resolution segmentation from these selected patches.

1.2 Aim of the Work

The primary objective of this work is to design and implement an informed, data-driven sampling method for segmentation of volumetric medical data. Such a method optimizes sub-volume selection to improve 3D anatomical shape reconstruction. Building upon the work of Lang et al. [LMA20], we propose and develop a deep learning-based sampling method, which processes the binary segmentation of medical shapes, and outputs shape-characterising proposals for volumetric patches (Figure 1.1). We integrate our proposed sampler into an end-to-end reconstruction pipeline, where the selected volumetric patches are provided as input to a pretrained state-of-the-art reconstruction network based on implicit neural representation [ALL+22], which maps the extracted sub-volumes to a full-resolution reconstruction of the input segmentation. Additionally, we aim to investigate the ability of our proposed learned sampler to sample shapes of varying complexity, as well as to assess the influence of the quality of the reconstruction network on the patches selected, to better understand the use cases where learned sampling is relevant.

Based on these objectives, the thesis answers the following research questions:

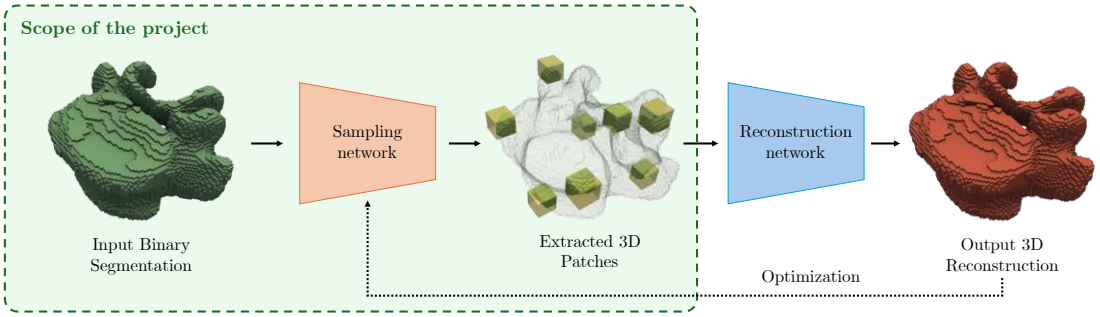


Figure 1.1: This thesis aims at developing a learned sampling strategy for extracting 3D patches from a binary segmentation. The proposed pipeline contains a sampling network (orange), producing patch proposals, and a subsequent reconstruction network (blue), processing extracted patches into full-resolution reconstruction. In our pipeline, the outputs of the reconstruction network are used to train the sampling network and for evaluation.

(RQ1) *Given a binarized 3D medical image, which patches are the most informative, such that using only the selected patches as the input to a reconstruction network will achieve the best performance based on selected quality metrics?*

Our working hypothesis is that reconstruction accuracy depends strongly on the selected regions. Patches sampled from areas containing relevant anatomical structures are expected to provide more useful information than those taken from homogeneous background regions. In this research question, we aim to investigate how the proposed sampler can identify and prioritize the most informative patches, ensuring that the reconstruction network achieves high performance across selected quality metrics, and how these patches compare to those selected by baseline methods.

(RQ2) *To what extent does the selected patch configuration, i.e., the patch-size and the number of patches, affect the reconstruction quality?*

The patch size and the number of patches represent the amount and distribution of the context provided to the reconstruction network. Although larger patches provide a broader local context, which can be beneficial for the reconstruction task, the additional context may be redundant or uninformative. Increasing the number of sampled patches allows the network to better position the samples around the target object, potentially improving coverage and reconstruction performance. In this research question, we aim to investigate the impact of patch size and the number of sampled patches on the accuracy and quality of the reconstruction.

(RQ3) *To what extent do the selected patch-size and number of patches affect memory*

requirements and inference time?

Larger patch sizes or an increased number of sampled patches potentially increase the memory requirements of the sampling module and lead to longer overall training times. In this research question, we investigate how the choice of patch size and number of patches impacts these computational aspects. Combined with the reconstruction performance results from RQ2, we analyze which configurations offer the best trade-off between reconstruction quality, memory usage, and inference speed.

1.3 Contribution

The main contributions of this thesis are summarized as follows:

- We adapt a sampling architecture based on point cloud representation to extract volumetric patches from binary segmentations of 3D medical images. The architecture is optimized to produce patches which, when used as input to a reconstruction network, maximize reconstruction accuracy. To this end, we introduce a deep learning pipeline consisting of (i) an adapted point cloud sampling architecture, (ii) conversion modules that transform data between voxel grid and point clouds representations, and (iii) a state-of-the-art reconstruction network that reconstructs the full volume from sampled patches. We optimize the sampling architecture using an existing loss function, which we adapt to better suit the reconstruction of volumetric medical images.

- We perform a comprehensive evaluation of our proposed pipeline, demonstrating that the learned sampling strategy consistently outperforms selected non-learned baselines, achieving an improved reconstruction DICE Score of 1-5%. The proposed method shows the largest improvements when (i) sampling shapes with complex geometries, (ii) the modeling power of the reconstruction model is poor, and (iii) in scenarios where only a limited spatial context is available. Furthermore, we investigate the trade-off between patch size and the number of sampled patches, identifying configurations that improve accuracy or reduce memory and computational cost.

- We demonstrate that the additional temporal and memory overhead introduced by the sampling and reconstruction stages is small compared to the savings achieved by processing only a subset of the volume. By strategically selecting patches for processing, our pipeline significantly reduces the volume of data handled by expensive operations, enabling faster processing and deployment in resource-constrained environments.

1.4 Outline of the Thesis

The contents of this Thesis are structured as follows: Chapter 2 reviews related work, covering reconstruction from sparse samples in medical data as well as influential approaches to data sampling across different data modalities. In Chapter 3, we present our proposed sampling pipeline designed to strategically select patches for volumetric reconstruction. Chapter 4 details the experimental setup, including experimental pipeline, benchmarking datasets with the applied preprocessing operations, evaluation protocols, implementation details, and selected metrics. In Chapter 5, we additionally provide a discussion of the results, including evaluation metrics, performance measures, qualitative evaluation of predicted samples and their respective reconstructions for different datasets, sampling ratios, and patch sizes. Finally, Chapter 6 summarizes the main contributions

of the thesis, presents comprehensive answers to the research questions, and discusses limitations and potential future work.

CHAPTER 2

Related Work

The contents of this chapter are split into two sections: reconstruction of medical data from sparse measurements and data sampling. This chapter split is motivated by the focus of this work on informed 3D sampling of medical data, in which the sampling strategy is guided by a reconstruction task. For this reason, we first review reconstruction methods in Section 2.1, which define how volumetric data can be represented and how shared object priors can be learned to enable the reconstruction of unseen volumes from limited observations. Afterwards, Section 2.2 covers methods how to strategically sample subsets of the input, which maximize the accuracy of the reconstruction. Since the proposed sampling pipeline converts volumetric data into a point cloud representation, where the actual sampling is performed, Section 2.2 reviews primarily state-of-the-art sampling methods operating on point clouds.

2.1 Reconstruction of Medical Data from Sparse Measurements

Reconstructing medical data from sparse measurements is a critical step in enabling accurate analysis while reducing acquisition costs and time [JMFU17, SCH+18, LYY17]. Section 2.1.1 presents implicit volumetric representation, which defines continuous volumetric functions over 3D space, as an alternative to conventional explicit representations that describe data using discrete structures. Such representations are particularly relevant for reconstruction tasks, as they allow learning a shared object prior and restoring volumetric structures from limited measurements [TTM+22, ALL+22]. Section 2.1.2 reviews medical data reconstruction methods based on statistical modeling and convolutional neural networks, and shows how neural networks can learn continuous implicit representation of volumes, enabling high-resolution reconstructions through continuous sampling. The class of neural networks that learn shape representations via implicit functions is particularly relevant to our research, as they enable the reconstruction of full-resolution shapes from samples with continuous coordinates, without being constrained by voxel

grids or discrete sampling. Furthermore, Section 2.1.2 introduces the reconstruction network utilized in the proposed sampling pipeline to provide a reconstruction loss, which guides the training of the learned sampler. By leveraging implicit volumetric representations, the sampler can generate points at arbitrary continuous locations, which can be directly processed by the reconstruction network to guide reconstruction-driven training.

2.1.1 Representation of Volumes

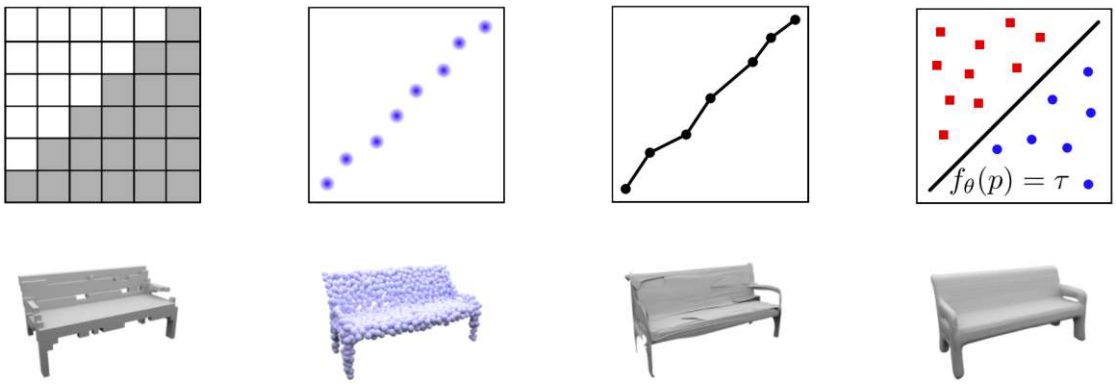
A conventional approach to encode a 3D volume is referred to as explicit representation, which stores volumetric information in discrete data structures such as voxel grids, point clouds, or surface meshes, enabling straightforward access to individual elements (Figures 2.1a-c), direct visualization, and integration with a standard graphics rendering pipeline [TTM+22]. However, these formats are inherently limited in resolution due to discretization, and each comes with additional drawbacks: voxel grids suffer from high memory demands that scale cubically with resolution; point clouds lack inherent connectivity, requiring additional processing to recover surface or volume structure; and mesh-based approaches are often constrained by the use of template deformation, which limits their ability to represent arbitrary topologies [MON+19].

More recently, alternative approaches emerged, using implicit representation (Figure 2.1d) to model volumetric data as continuous functions that map spatial coordinates to scalar values, which allows sampling at arbitrary resolution [PFS+19, ALL+22]. However, when applied to reconstruction tasks, the flexibility of implicit representations comes at the cost of increased computational complexity, as the underlying function is defined over a continuous domain and must be densely evaluated on a discrete coordinate grid to produce explicit outputs such as segmentation masks or volumetric reconstructions [ALL+22].

A common implicit representation function for surface representation is the *Signed Distance Function* (SDF) [CZ19, PFS+19, SMB+20, TTM+22]. Assuming a well-defined and watertight surface boundary, the SDF defines a continuous mapping from spatial coordinates to the distance to the closest surface, where the sign indicates whether a point lies inside (negative) or outside (positive) the surface (see Figure 2.2). Let Ω be a subset of a metric space X with metric d, and $\partial\Omega$ be its boundary. Then, SDF is defined as:

$$f_{\text{SDF}} : \mathbb{R}^3 \to \mathbb{R}, \quad f_{\text{SDF}}(\mathbf{x}) = \begin{cases} d(\mathbf{x}, \partial\Omega), & \text{if } \mathbf{x} \in \Omega \\ 0, & \text{if } \mathbf{x} \in \partial\Omega \\ -d(\mathbf{x}, \partial\Omega), & \text{if } \mathbf{x} \notin \Omega \end{cases}$$

Complementary to SDFs, which represent surfaces, *occupancy functions* [MON+19] provide an alternative implicit representation focused on modeling volumes, and are defined as:



(a) Voxel (b) Point (c) Mesh (d) Implicit Function

Figure 2.1: 3D data representation: (a) voxel-based representation, (b) point-based representation, (c) mesh-based representation, (d) representation using a continuous implicit function f_θ, where the decision boundary defines the surface of the object. Image adapted from Mescheder et al. [MON+19]

.

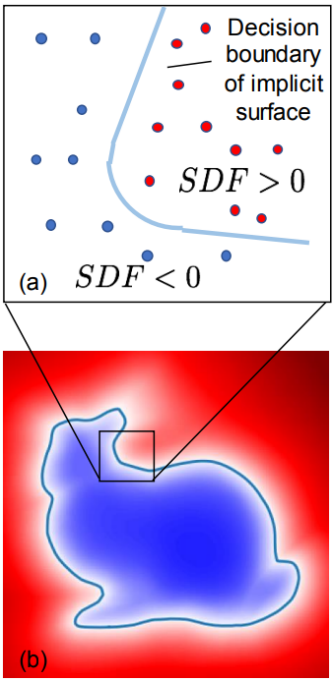


Figure 2.2: Implicit surface representation using the Signed Distance Function. Image by Park et al. [PFS+19]

$$f_{\text{occupancy}} : \mathbb{R}^3 \to \{0, 1\}, \quad f_{\text{occupancy}}(x) = \begin{cases} 1 & \text{if } x \in \Omega, \\ 0 & \text{if } x \notin \Omega, \end{cases}$$

The occupancy function maps each spatial point $x \in \mathbb{R}^3$ to a binary value, reflecting its spatial relation to the subset Ω, assigning 1 to points inside Ω and 0 to those outside.

Implicit functions can be used to represent surfaces and volumes of varying complexity. For simple scenes, e.g., a plane, a linear implicit function suffices [TTM+22]. To handle more complex surfaces or volumes, polynomials [SAG84], multivariate Guassians [WF06] or Gaussian mixtures [HHGCO20] can be used. Alternatively, Multi-Layer Perceptrons (MLPs), which are known to act as Universal Function Approximators [HSW89], can be trained to learn the underlying implicit representation function [TTM+22].

Mescheder et al. [MON+19] observe that the occupancy function $f_{occupancy}$ can be approximated by a neural network. They further propose conditioning the network on an additional input that encodes a specific shape instance, allowing the model to learn a shared shape prior across a population of objects and to reconstruct multiple shapes using a single set of network parameters. The proposed *occupancy network* takes a pair $(\mathbf{x}, c) \in \mathbb{R}^3 \times \mathcal{C}$, where $\mathbf{x}$ denotes a spatial coordinate and $c \in \mathcal{C}$ is a conditioning variable that represents the shape-specific information, and outputs a scalar corresponding to the occupancy probability.

The continuous formulation of implicit functions allows the representation of shapes and volumes at arbitrary spatial resolution, which is well-suited for capturing fine geometric details [MON+19, PFS+19]. Additionally, implicit representations are more memory-efficient than explicit alternatives such as voxel grids or point clouds, since shapes of different resolutions are encoded in the parameters of the implicit function [MON+19, TTM+22]. Another benefit is the flexibility: the same network can represent a wide range of shapes by conditioning on different inputs, enabling generalization across shape populations [BJLPS18, MON+19, ALL+22]. As a result, implicit functions are well-suited for capturing medical data shape populations, especially for representations with varying voxel spacings and resolutions [TTM+22, ALL+22].

Implicit neural representations with shape-specific conditioning are widely adopted for reconstructing shapes or volumes from sparse or partial observations. This is typically achieved through inference-time optimization, where either a latent vector [BJLPS18, ALL+22] or the network parameters [SCT+20, OB22, LTLS21] are adapted based on the collected samples. Once the shape-specific condition is obtained, the full-resolution object can be inferred. However, this inference-time optimization can be computationally expensive and time-consuming [PFS+19].

While the use of implicit functions for shape reconstruction presents certain drawbacks, such as the need for sampling using a dense coordinate grid or inference-time optimization, these are outweighted by their advantages. The ability to capture fine details, operate across different resolutions and modalities, and to adapt to different voxel sizes and

spacings makes them particularly suitable for modeling of medical data. For this reason, this work employs learned representations of medical data using implicit functions and further improves reconstruction performance by learning to assess the relevance of the input to the reconstruction task.

2.1.2 Medical Data Reconstruction

This section reviews influential approaches for reconstructing medical data from sparse measurements. In the proposed pipeline, reconstruction is a key component during sampler training, as it provides a training signal to the supervised sampler through gradient backpropagation. Additionally, it serves as a means of quantitative evaluation and comparison between different sampling strategies.

Adler et al. [AÖ17] define the reconstruction as an inverse problem. Inverse problems refer to problems where one seeks to reconstruct parameters characterizing the system under investigation from indirect observations [AÖ17]. Mathematically, an inverse problem can be formulated as reconstructing (estimating) a signal $f_{\text{true}} \in X$ from data $g \in Y$ such that:

$$g = \mathcal{T}(f_{\text{true}}) + \epsilon, \tag{2.1}$$

where X and Y are topological vector spaces, $\mathcal{T} : X \to Y$ (forward operator) models how a given signal gives rise to data in the absence of noise, and ϵ is a random noise [AÖ17]. When data samples $g \in Y$ are undersampled and noise corrupted, the inverse problem in Equation 2.1 is ill-posed [MTMHS21]. As a result, a direct inversion of $\mathcal{T}$ is generally not possible [MTMHS21].

Medical data reconstruction has seen significant advancements in recent years, evolving from Classical Approaches that capture the shape variability using low-dimensional embeddings [CET01, MBL+02, DYY+17], to convolutional neural networks (CNNs) that exploit spatial locality and hierarchical features [JMFU17, SCH+18, KMY17, WTH+19], to implicit neural representations capable of representing complex anatomical structures in a continuous and memory-efficient manner [BJLPS18, PFS+19, ALL+22]. The following sections review selected influential methods from each category.

Classical Approaches

Classical Approaches aim to learn a low-dimensional embedding of medical data by capturing patterns and variability across a dataset [DYY+17]. A common building block utilized by Classical Approaches is the Principal Component Analysis (PCA), which decomposes data into orthogonal basis vectors (principal components) to enable dimensionality reduction. When applied to a dataset of medical shapes, it can extract a compact, low-dimensional representation that approximates the original data with minimal loss. PCA can be utilized, for example, in Active Shape Models (ASM) to capture shape variation within a population. These models represent anatomical structures by

learning a statistical distribution over aligned landmarks in 2D [CET01], or 3D [MBL+02]. Additionally, the PCA basis itself can be used as a projection space for new data, as is the case in Dietz et al. [DYY+17], who learn the variation of lungs from MRI images using PCA. Specifically, they first collect a set of fully-sampled lung MRI scans and compute a PCA basis. When a new scan is acquired with missing or undersampled measurements, this incomplete data is projected onto the learned PCA basis and the resulting PCA coefficients are used to reconstruct the high-resolution image as a weighted sum of the basis components.

However, Classical Approaches come with several limitations. As they are primarily designed to capture global or dataset-wide properties, such approaches often fail to represent local variations [LWW+11]. For instance, Classical Approaches perform well for modeling datasets of healthy organs, but struggle with small pathologies or anomalies that were absent or underrepresented in the training data [MCAL16]. Additionally, methods like the PCA assume linear relationships within the data, which restricts their ability to represent highly non-linear or irregular anatomical deformations [MWL+17].

These limitations have motivated a shift towards deep-learning-based approaches, capable of capturing both global and local structures with fine detail, including complex and non-linear anatomical variations with more effective generalization across diverse cases [JMFU17, SCH+18].

CNN-based Models

In recent years, deep learning has transformed medical image reconstruction by enabling models to learn complex inverse mappings directly from data [JMFU17, SCH+18]. Instead of relying on handcrafted priors or low-dimensional shape models, deep learning models are trained on large datasets to learn reconstruction functions that generalize across different anatomical structures, data modalities, and acquisition conditions [JMFU17]. In particular, CNNs have proven effective due to their ability to capture both local and global image context using hierarchical, multiscale feature extraction, enabling tasks such as reconstruction from limited observations [JMFU17, SCH+18], learning denoising of corrupted images [KMY17], super-resolution [WTH+19], or segmentation [ZRSTL18]. We present related work in CNN-based medical image reconstruction, which specifically focuses on reconstruction from sparse or undersampled data. The selected studies address the challenge of reconstructing a full-resolution 2D or 3D medical data from partially observed measurements, which directly aligns with our research.

A notable contribution comes from Jin et al. [JMFU17], who address the challenge of accurate reconstruction of high-accuracy CT-scans from subsampled sinograms, which represent projection data acquired at different viewing angles during the CT scanning process. By reducing the number of scanning angles, this approach aims to lower radiation exposure and acquisition time while maintaining reconstruction accuracy. However, subsampled scans might lead to reduced image quality and could lead to image artifacts [JMFU17]. Proposed FBPConvNet (Figure 2.3) learns to remove artifacts and restore fine details in

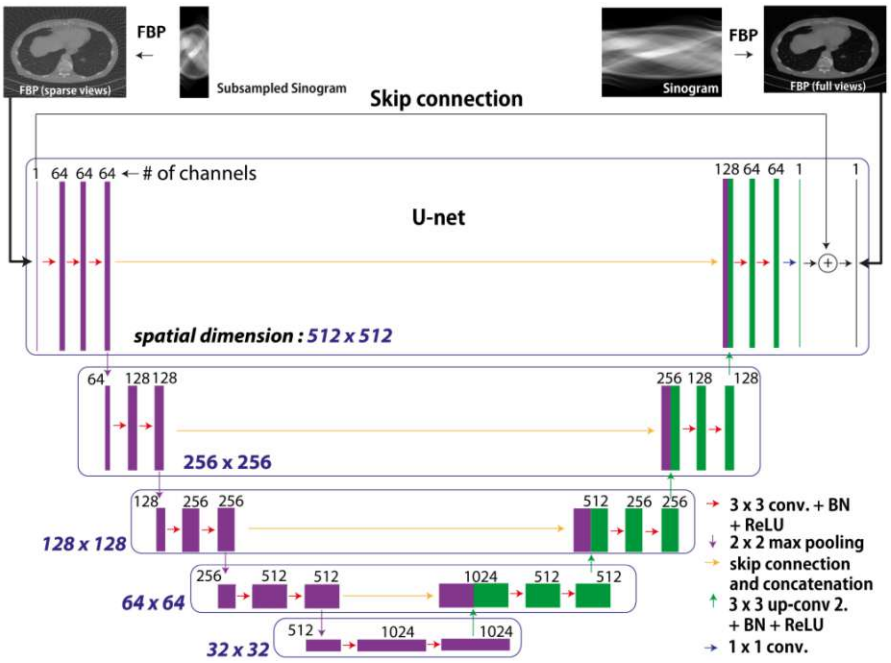
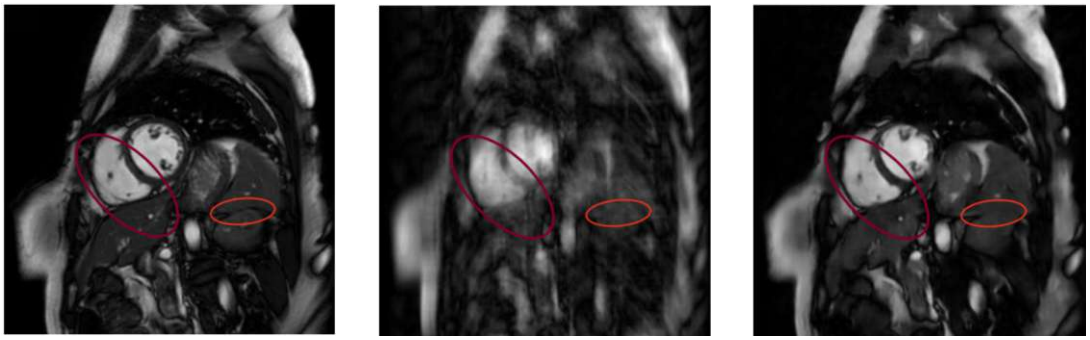


Figure 2.3: Architecture of FBPConvNet [JMFU17]. Subsampled sinograms are projected into a low-resolution spatial domain with artifacts. FBPConvNet is then trained to restore the high-resolution equivalent. Image by Jin et al. [JMFU17].

the reconstructed images. The architecture, based on U-NET [RFB15], uses hierarchical downsampling and upsampling layers with skip connections to capture both local and global features, which effectively removes artifacts and restores fine image details.

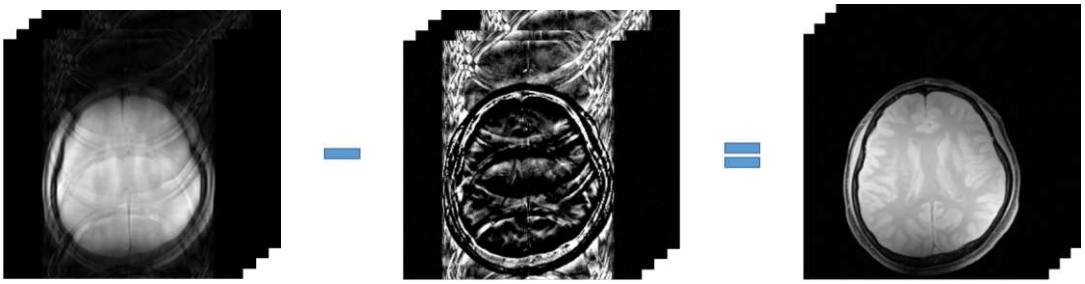
Schlemper et al. [SCH+18] tackle the challenge of accelerating MRI acquisition to reduce scan times and improve patient comfort. An output of an accelerated MRI is a sparsely sampled k-space, where a reduced number of encoding steps leads to missing lines, which are typically set to zero. Transforming subsampled, zero-filled k-space to the image domain leads to artifacts, such as those depicted in Figure 2.4b. Schlemper et al. [SCH+18] propose a network, which learns to remove the artifacts and predicts an artifact-free MRI scan (see Figure 2.4c). The proposed architecture consists of a cascade of CNN blocks interleaved with Data Consistency layers, which ensure that the network preserves the originally acquired k-space values while progressively removing artifacts.

Similarly to Schlemper et al. [SCH+18], Lee et al. [LYY17] tackle the challenge of accelerating MRI scanning by reconstructing artifact-free images from sparsely sampled k-space data. However, compared to conventional approaches, such as Schempler et al. [SCH+18] or Jin et al. [JMFU17], the proposed method learns to predict aliasing artifacts rather than the aliasing-free fully sampled reconstruction. The aliasing-free reconstruction is then obtained by subtracting the corrupted image from the estimated aliasing artifact (see Figure 2.5).



(a) Full resolution (b) 3x subsampled (c) Reconstruction

Figure 2.4: (a) MRI scan obtained from full-resolution k-space, (b) MRI scan obtained from k-space, which was 3x subsampled, (c) reconstruction from the subsampled scan from Schlemper et al. [SCH+18]. Image by Schlemper et al. [SCH+18].



Input image with artifacts | Estimated aliasing artifact | Artifact-free image

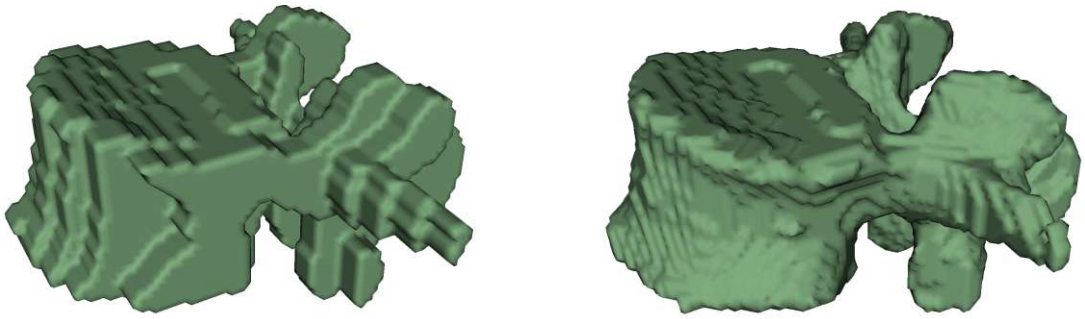
Figure 2.5: Reconstruction approach by Jin et al. [JMFU17] learns to estimate the aliasing artifact, which is then subtracted from the input corrupted image. Image by Jin et al. [JMFU17].

However, CNN-based sampling methods, along with Classical Approaches, share some common limitations: first, their resolution is limited to the resolution of training data, and second, they are fixed to a single discretization and, thereby, require resampling of volumes with variable spacings to a common one [ALL+22]. In comparison, INR models learn a continuous representation of the shape population, allowing sampling shapes at arbitrary resolution during both training and inference [ALL+22].

Implicit Neural Representation Models

Implicit Neural Representation (INR) Models use neural networks to learn a continuous implicit function that represents image or volume data [PFS+19, ALL+22]. This continuous characteristic enables reconstruction at arbitrary resolutions, while the neural network parameterization provides a memory-efficient and compact representation [MAT+23].

Wu et al. [WLX+21] tackle the challenge of reconstructing high-resolution 3D MRI scans



(a) Segmentation from thick slice MRI with anisotropic voxel size 1×1×6 mm.

(b) Segmentation from thin slice MRI with isotropic voxel size 1×1×1 mm.

Figure 2.6: Visualization of vertebrae segmentation using thick slices (a), and after high-resolution reconstruction (b).

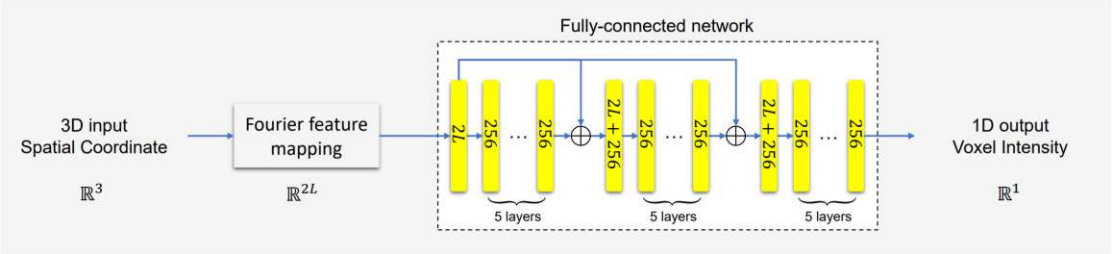


Figure 2.7: Architecture of IREM [WLX+21]. Image by Wu et al. [WLX+21].

from low-resolution images. Oftentimes, MRI scans are acquired with slice thicknesses of several millimeters (so-called thick slices) to shorten the acquisition times, improve patient comfort, and reduce motion artifacts [TBO+21]. However, such scans have high in-plane resolution, but poor resolution along the depth dimension, resulting in anisotropic volumes (see Figure 2.6 for example) [TBO+21].

Proposed architecture IREM (depicted in Figure 2.7) learns to reconstruct a high-resolution 3D MRI scan from a stack of thick-slice, low-resolution images acquired in multiple orientations. IREM takes as input a 3D spatial coordinate and predicts an intensity value. To enable the network to learn high-frequency features from the low-resolution stacks, the input 3D coordinate is first projected to a higher-dimensional space using Fourier feature mapping. Afterwards, the projected coordinate is fed to an MLP with ReLU activations, which produces the intensity prediction. The network is first optimized using the coordinate-intensity pairs from the low-resolution input. Afterwards, a high-resolution reconstruction is obtained by sampling the model using dense grid coordinate samples.

Sitzmann et al. [SMB+20] highlight the problem of conventional neural representation networks consisting of an MLP with ReLU activation. Because ReLU networks are piecewise linear, they lack the capacity to represent fine details and typically do not

capture the derivatives of the target signal well. To address these limitations, Sitzmann et al. [SMB+20] propose SIREN, an MLP network leveraging the periodic sine function as the activation.

MLPs with periodic activations are further utilized by Shen et al. [SPX24] to train a patient-specific anatomy prior from a single full-resolution scan, which then serves as an initialization to reconstruct follow-up scans from sparse measurements. Proposed architecture, NeRP, maps continuous spatial coordinates to image intensities and learns a compact, high-resolution representation of individual patients' anatomy. A key advantage of this approach is that supervision is inherently available from the full-resolution scan itself, eliminating the need for manual annotations or large-scale datasets. Similarly to other neural implicit representation methods, NeRP can represent data of arbitrary resolution. However, due to the fact that the prior must be trained separately for each individual, the method lacks the ability to generalize across patient populations.

The concept of representing implicit functions as occupancy networks is further investigated by Amiranashvili et al. [ALL+22], who apply it to the reconstruction of medical shapes from sparse segmentations (Figure 2.9). We discuss this method in more detail because it is a promising approach specifically designed for reconstructing voxel grids of medical shapes from sparse binary segmentations, which directly aligns with our goals. For this reason, we adopt this network in our proposed pipeline.

In order to model variations among medical shapes, latent vectors $z \in \mathbb{R}^d$ are used as shape-specific embeddings to condition on. The proposed MLP f_θ, mapping the underlying implicit function, has, therefore, the following functional form:

$$f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \rightarrow [0, 1]$$

Following Bojanowski et al. [BJLPS18] and Park et al. [PFS+19], Amiranashvili et al. [ALL+22] employ an auto-decoder architecture with a sigmoid activation as f_θ. At the start of training, a randomly initialized latent vector $z \in \mathbb{R}^d$ is assigned to each data sample. During training, individual latent vectors are jointly optimized with the parameters of f_θ. For every training volume i and voxel j, exact voxel coordinates $x_i^j \in \mathbb{R}^3$ and the corresponding ground-truth occupancy values $y_i^j \in \{0, 1\}$ are extracted. For a batch of randomly chosen voxels, a set of predicted occupancy values is obtained. The difference to the ground-truth occupancy is minimized through a loss function, consisting of volume-wise soft Dice Score and voxel-wise binary cross-entropy (Figure 2.8).

The inference of a full-resolution reconstruction of the medical shape from sparse segmentation $\mathcal{S}$ happens in two steps. First, the latent vector corresponding to the sparse input $\mathcal{S}$ is determined. Afterwards, the obtained continuous occupancy function is sampled at the target resolution to obtain the voxel-based reconstruction. The latent vector that corresponds to a given observation $\mathcal{S}$ is first randomly initialized and then optimized through gradient descent. Similarly to the training procedure, the MLP f_θ is sampled at positions of voxels from observation $\mathcal{S}$, and the same loss function is used as the

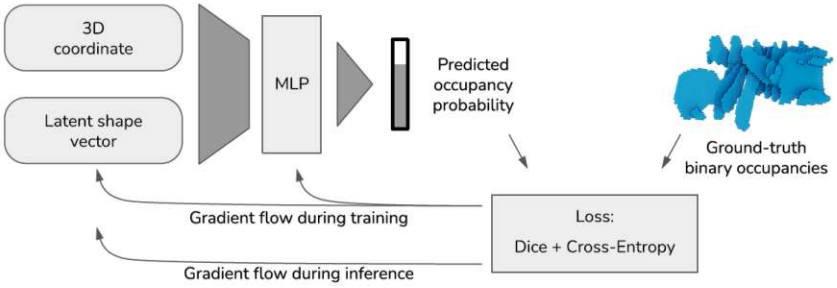
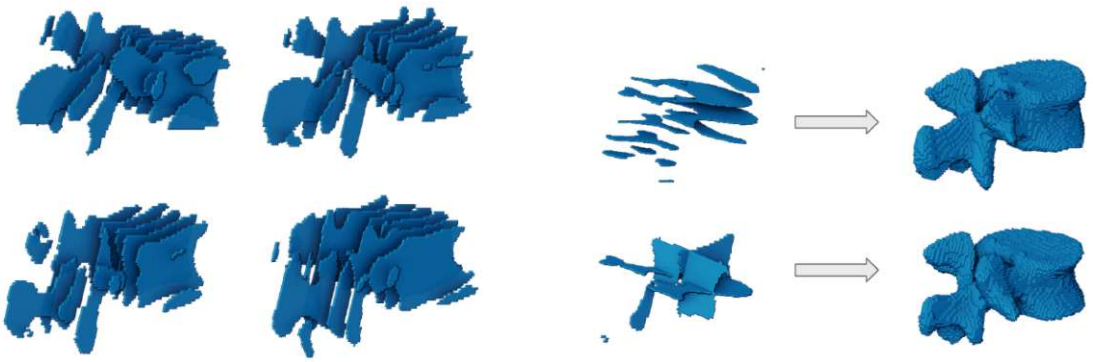


Figure 2.8: Pipeline of the method proposed by Amiranashvili et al. [ALL+22]. *Training*: both MLP and latent vector are optimized. *Inference*: MLP remains fixed and only latent vector is optimized. Image by [ALL+22].



(a) Sagittal training data examples (b) Reconstruction from different inputs

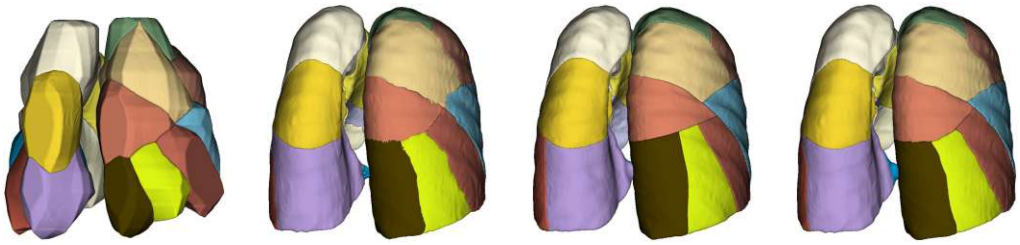
Figure 2.9: Amiranashvili et al. [ALL+22] train a neural network capturing a shared prior over medical shape segmentations. The architecture is trained from sparse slices (a), but during inference, it generalizes to arbitrary slice configurations (b). Image by Amiranashvili et al. [ALL+22].

minimization objective. The difference from the training phase is that the weights θ of the MLP f_θ are frozen, and only the latent vector z is optimized (Figure 2.8). The loss is minimized with respect to all voxel positions and their occupancy values $(x_k, y_k) \in \mathcal{S}$:

$$z_{rec} = \arg\min_z \sum_k \mathcal{L}(f_\theta(x_k, z), y_k)$$

Using the optimized latent vector z_{rec}, the occupancy function $f_\theta(\cdot, z_{rec})$ can be sampled at any desired resolution and spacing to obtain the high-resolution voxel-based reconstruction.

In a subsequent extension, Amiranashvili et al. [ALL+24] further improve their framework by incorporating pose augmentations to reconstructed shapes through random affine transformations, improving the robustness and generalization abilities of their INR



(a) Pre-trained Template (b) After Deformation (c) After Correction (d) Ground Truth

Figure 2.10: Template-guided pulmonary reconstruction: a shared template (a) is deformed to match patient anatomy (b), refined by a correction network (c), and compared to the ground truth segmentation (d). Image by Xie et al. [XZK+26].

network. Additionally, the authors analyze the learned d-dimensional latent space and demonstrate its desirable properties. By training a linear SVM on manually defined classes and performing PCA on the latent vectors, they show that the latent space is linearly separable with respect to defined classes, captures meaningful anatomical variations, and is smooth and continuous. This allows to generate new valid shapes through latent space sampling or perform interpolation of latent vectors to produce smooth transitions between anatomies.

Xie et al. [XZK+26] tackle problem of segmentation of pulmonary segments. The authors argue that the quality of segmentation does not depend solely on the voxel-level prediction quality but also on anatomical correctness. For this reason, instead of employing per-voxel segmentation network, they authors rephrase the task as a reconstruction of individual pulmonary segments. To achieve this, the proposed method leverages a template-guided neural implicit framework, formulating a continuous occupancy function f that maps any 3D point $p \in \mathbb{R}^3$ in patient space to a probability distribution over pulmonary segment classes:

$$f(p) = T(p + \Delta d(p)) + \Delta c(p),$$

where $T \in \mathbb{R}^{D \times H \times W \times 19}$ is a pretrained voxel-grid template with multi-class occupancy values (Figure 2.10a), evaluated at the deformed point $p + \Delta d(p)$ using trilinear interpolation, $\Delta d(p) \in [-1, 1]^3$ is a displacement vector predicted by a deformation network, shifting the patient-space point p into the template space (Figure 2.10b), and $\Delta c(p) \in \mathbb{R}^{19}$ is a local correction term, predicted by a correction network to refine the output distribution and refine anatomical details (Figure 2.10c). The use of templates ensures the produced reconstruction matches the global shape of the pulmonary segments, while the implicit formulation of f ensures the segmentations can be evaluated at arbitrary resolution.

The outlined recent work shows that INRs are a prominent alternative to conventional data representation, offering benefits in continuous formulation, ability to avoid locality biases,

and differentiability [MAT+23]. Examples discussed above show that INR networks are capable of capturing different types of medical data (binary segmentation in [ALL+22], image intensity in [SPX24, WLX+21]), learn shape priors over populations of different sizes (single shape in [WLX+21], single-patient population in [SPX24], large-scale datasets in [ALL+22]), and are able to reconstruct high-resolution data from sparse samples ([ALL+22, WLX+21, SPX24]). In this work, the architecture proposed by Amiranashvili et al. [ALL+22] is further adopted, as it targets the reconstruction of segmentations and captures a shared shape prior across the entire dataset instead of relying on patient-specific information.

2.2 Data Sampling

Data sampling refers to a process of selecting a subset of the input data for further processing [DLA19, LMA20]. This substantially decreases the computational and memory requirements of the whole pipeline compared to processing a full-scale input, especially with volumetric inputs.

An example of a task that benefits from data sampling is volumetric segmentation, which has numerous applications such as tumor detection [RFB15], volume estimation [GTQ+18], or anatomical structure delineation [WBM+23]. To perform volumetric segmentation efficiently, various modeling strategies have been developed that balance accuracy with computational cost. 2D models decompose the 3D volume into a batch of 2D slices and make predictions independently for each slice [LXZ+18, WCW+22, KJI+24]. While 2D models are computationally efficient, they fail to capture inter-slice relationships, potentially reducing the accuracy of the final whole-volume prediction [JYF+25]. In comparison, 3D models analyze the complete volumetric input without any decomposition, including inter-slice relationships, resulting in improved accuracy [LLX+22, AHL+23]. However, this structural fidelity comes at the expense of significantly higher computational complexity and memory requirements [JYF+25]. A potential solution is to use a sliding window approach, which samples a set of significantly smaller volumetric patches at uniform intervals with some degree of overlap. Each patch is then processed separately, and the overlapping predictions are aggregated into the final volume. Even though the sliding window approach is more memory-efficient and allows processing volumes of large sizes, it suffers from a slower inference speed and increased computational demands [JYF+25]. This is because the sliding window approach does not rank patches according to their importance and segments all patches with equal computational resources, even those that clearly do not contain objects of interest or the object is simple enough for a less complex model to handle equally well [JYF+25]. Using an informed sampling approach has the potential to identify patches containing objects of interest and better distribute the effort and resources required for the task.

This work proposes a volumetric sampling pipeline that simplifies the sampling step by performing it directly on point clouds. The pipeline begins by converting the original volumetric voxel representation into the point cloud domain, from which the sampler

selects a representative point subset. Afterwards, volumetric patches are extracted by mapping the selected samples back to volumetric space.

The Section 2.2.1 reviews influential sampling methods based on point clouds. In Section 2.2.2, this work identifies and highlights key methodological differences among sampling methods, which serve to organize and compare existing approaches based on their core design choices. Section 2.3 highlights both strengths and weaknesses of the discussed sampling methods and identifies several open challenges, which this work is addressing.

2.2.1 Point Cloud Sampling

This category of sampling networks operates on datasets consisting of point clouds, where each element is a 3D point $p \in \mathbb{R}^3$. Given an original point set $P \in \mathbb{R}^{n\times 3}$ and a sample size $k \leq |P|$, the objective of the sampling operation is to select a subset $S \subseteq P, |S| = k$, which serves as a representative simplification of the original set. Due to the unordered nature of point clouds, these networks are typically designed to be permutation-invariant and robust to varying input point cloud sizes [LMA20, YSA23].

Dovrat et al. [DLA19] define point cloud sampling as a minimization problem: given a point set $P = \{p_i \in \mathbb{R}^3, i = 1, ...n\}$, a sample size $k \leq |S|$ and a downstream task network f_{task}, the goal is to find a subset S^* of k points that minimizes the task network's objective function f_{obj}:

$$S^* = \arg\min_S f_{\text{obj}}(f_{\text{task}}(S)), S \subseteq P, |S| = k \leq |S|.$$

Training point cloud sampling models poses a challenge of selecting actual subsets from the input point cloud while maintaining differentiability, which is essential for enabling backpropagation and end-to-end model training [DLA19]. Some approaches solve this by generating a set $\mathcal{G} = \{g_1, ..., g_k\}, g \in \mathbb{R}^3$, and optimize the network either using the generated samples [DLA19], or through soft-projection [LMA20], which projects the generated points onto the input point cloud in a soft and differentiable manner. During inference, these methods employ non-differentiable matching to obtain the set S from the generated set $\mathcal{G}$. Other methods define sampled sets as weighted linear combinations of the input points [YYJ22, YSA23].

Approaches to point cloud sampling can be divided into learned and non-learned. Non-learned methods select representative subsets based on geometry heuristics, without considering any relevance to the downstream task objective [DLA19]. An example of a non-learned sampling method is the Farthest Point Sampling (FPS) algorithm [ELPZ97, MD03]. FPS iteratively selects points that are farthest from the already selected subset, which maximizes spatial coverage. In comparison, learned sampling approaches exploit task-specific supervision by utilizing gradients of the task network, enabling the selection of points with the most significant effect on the objective of the task network.

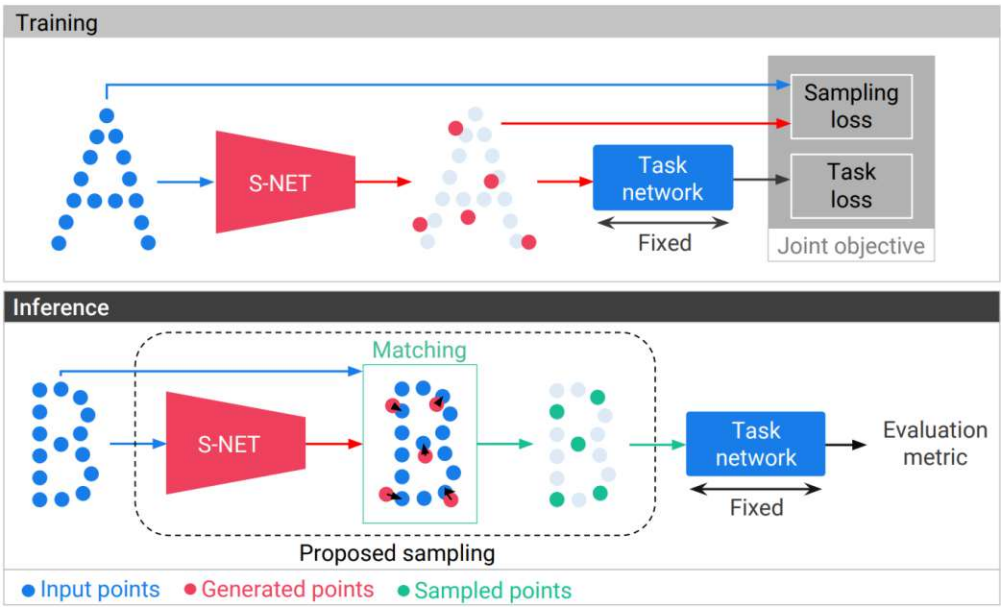


Figure 2.11: Overview of the S-NET sampling network. Training phase: S-NET-generated points are passed to a pre-trained and fixed task network. The loss of the task network is combined with a simplification loss of the sampler to form a minimization objective. Inference phase: generated points are matched with the input point cloud to get a subset of it. These points are then fed to the reconstruction network for evaluation. Image by Dovrat et al. [DLA19].

Dovrat et al. [DLA19] propose S-NET, a point cloud simplification network based on the PointNet [CSKG17] architecture. S-NET takes a set P of 3D coordinates representing a 3D shape, and generates a set S^* of k samples, representing a simplified shape. S-NET works in tandem with a task network, which is pretrained on a point-cloud-based task (e.g., classification, reconstruction, retrieval). The task network remains fixed during S-NET training, ensuring that sampling is optimized to the task rather than the task being optimized to an arbitrary sampling.

S-NET is operating on a set of unordered point clouds, which undergo a series of 1×1 convolutional layers. Each point is, therefore, treated separately from other points, which at the end yields a feature vector for each input point. To arrive at a global description vector, Dovrat et al. [DLA19] propose utilizing a max-pool operation. Finally, a series of fully connected layers predicts a set of 3D coordinates.

Due to S-NET predicting a set of continuous 3D coordinates in space, it is not guaranteed that this generated set will be a subset of the input. To overcome this limitation, Dovrat et al. [DLA19] propose a two-phase approach: during the training of the sampler, the task network receives generated coordinates from the sampler, while at the inference time, generated coordinates will be matched to actual samples from the input point clouds using the non-differentiable nearest neighbour matching. Duplicate cases, where two

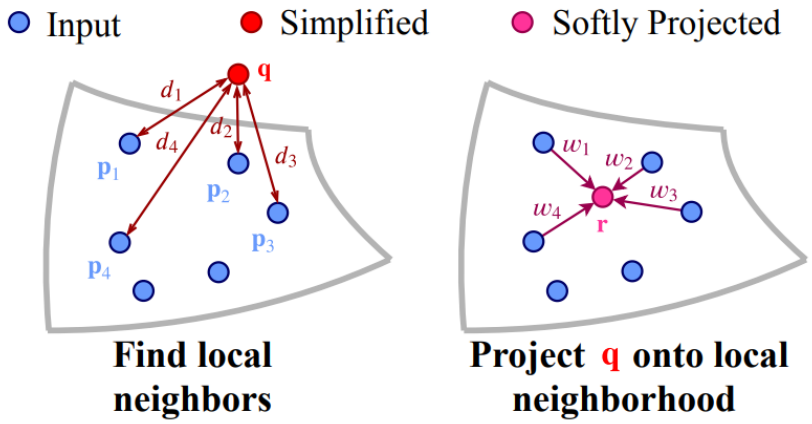


Figure 2.12: Soft-Projection module of the SampleNet network. For each simplified point q, its local neighbourhood is identified ($\{p_1, p_2, p_3, p_4\}$). Then, a soft projection r is obtained as a weighted average of the local neighbourhood. Image by Lang et al. [LMA20].

generated points would match the same point, are discarded, and missing samples are obtained using the FPS algorithm.

The combination of 1×1 convolution layers and feature-wise max-pool operation allows S-NET to process point clouds of arbitrary size. However, because the fully connected output layer has a fixed size, this network can only sample to a predetermined size, and changing it requires retraining the network from scratch. The S-NET network is optimized using a joint objective loss, formulated as the sum of a task loss and a sampling loss. The task loss is directly tied to the task network in use, whereas the sampling loss depends on the positions of the predicted samples. Figure 2.11 illustrates both the training and inference phases, and the joint objective loss.

While the S-NET effectively simplifies point clouds for downstream applications, the generated points are not guaranteed to be a subset of the input point clouds, as highlighted by Lang et al. [LMA20]. To address this issue, Lang et al. [LMA20] extend the S-NET architecture and propose SampleNet with a soft-projection operation, which is a differentiable approximation of sampled points as a mixture of points in the input point cloud (Figure 2.12). Each generated point $g \in \mathcal{G}$ is softly projected onto its neighbourhood, defined by k nearest neighbours in the complete point clouds. This yields a set of projected points $r \in R$, as a weighted average of original points from P. The proposed solution to the non-differentiable sampling problem consistently outperforms the S-NET architecture [DLA19] by a large margin in all benchmarked tasks, including classification, registration, and reconstruction. According to the authors, employing SampleNet to sample 32 points before PointNet saves about 90% of the inference time compared to applying PointNet to the complete point cloud, while requiring only an additional 6% more memory and incurring a performance drop of less than 10%.

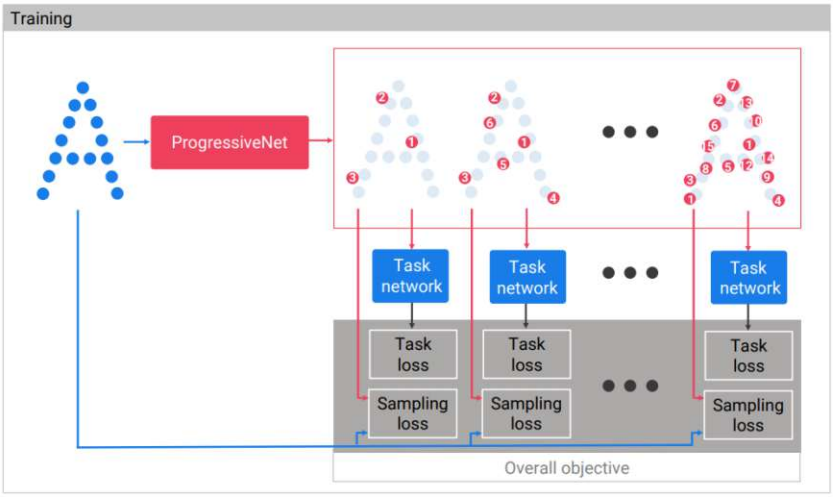


Figure 2.13: Hierarchical loss function allows training point cloud sampling networks to order output point samples according to their relevance. Increasingly larger subsets are treated as standalone samples and evaluated using the task network. Task and sampling losses across all subsets are summed to obtain the overall training objective. Image by Dovrat et al. [DLA19].

Both the S-NET and the SampleNet are limited by the size of the sampled subset. The last fully connected layer does not allow for flexible output sizes, and therefore each subset size requires a dedicated training run. To improve upon this limitation, Dovrat et al. [DLA19] propose to learn the ordering of points according to their relevance to the task network. The network is configured to output a point set with the same cardinality as the input. Then, a subset of a desired size is obtained by taking the first k points from the sorted output. To train the ordering, the output point set is divided into nested subsets of increasing size, starting with the first 2 points, doubling the subset size at each step. Each subset is then treated as a standalone sample and evaluated for both task loss and sampling loss. Finally, both loss terms for each subset size are summed up to create the overall network training objective (Figure 2.13). This training scheme encourages the most relevant points to appear earlier in the output, effectively ordering the input point cloud and allowing to sample arbitrarily sized point sets.

In contrast to S-NET and SampleNet, Yang et al. [YSA23] propose training in a self-supervised manner, without employing a task network during training. The sampling network is based on a modified version of the PointNet architecture [CSKG17], consisting of series of 1×1 convolutional layers. For each point, a single scalar score is predicted. Similarly to ProgressiveNet, this method predicts point samples in an ordered manner, allowing for the extraction of a flexible number of samples. A soft, differentiable sorting module based on Sinkhorn optimal transport [CTV19, XDC+20] permutes the input point cloud according to per-point scores. The network is optimized using a hierarchical contrastive learning module that processes progressively larger subsets of each point cloud

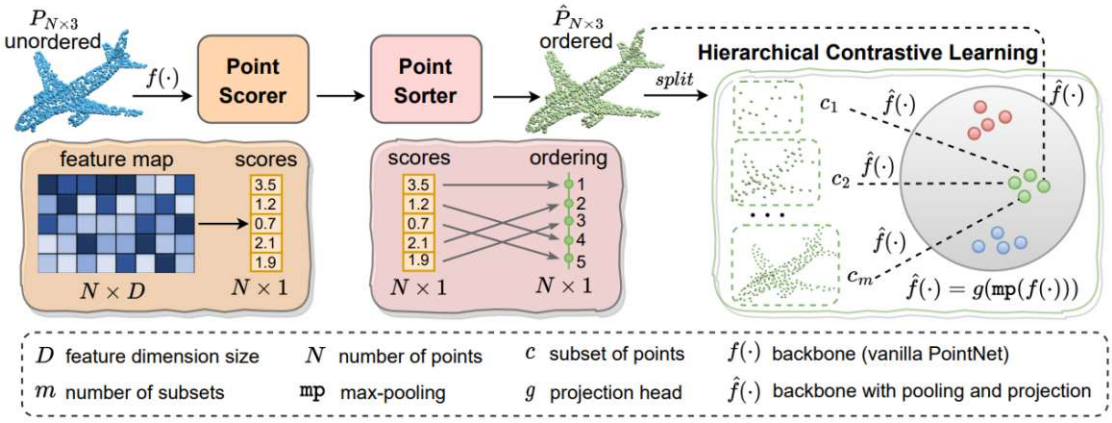


Figure 2.14: Point cloud sampling proposed by Yang et al. [YSA23]. A backbone produces per-point scores, which are used by a differentiable sorting module to sort the point cloud. The backbone is trained using a hierarchical contrastive learning module operating on subsets of different cardinality. Image by Yang et al. [YSA23].

and maps them onto a shared latent space. Within this space, representations of subsets originating from the same point cloud are encouraged to remain in close proximity, while projections from two different point clouds are pushed apart. Figure 2.14 illustrates the full pipeline.

Ye et al. [YYJ22] argue that the sampling process is naturally a sequential task, in which the points to be sampled next should depend on the points that have already been sampled. The proposed architecture, named APSNet, is based on the attention mechanism [VSP+17] and LSTM modules [HS97] to achieve a sequential point sampling scheme. The network processes an input point cloud of n points and generates m samples, which are obtained as a weighted sum of all n input points, and, therefore, are not guaranteed to be a subset on the input point cloud.

Similarly to S-Net [DLA19] and SampleNet [LMA20], APSNet utilizes a gradient from a pretrained and frozen task network during the training to optimize the sampling to a specific objective. LSTM modules allow the network to output samples in a sequential manner, taking into consideration previously generated points. First, all points are processed by a PointNet [CSKG17] feature extractor to obtain per-point feature vectors and a global context feature vector, which serves as an initial LSTM state. In each step, the LSTM uses the sampled point from a previous step to update its hidden state to reflect the history of all sampled points. Next, attention scores are computed between each per-point feature vector and the LSTM's hidden state to identify the next most informative point to sample. In order to maintain differentiability, output samples are generated as a weighted sums of attention scores and all n points in the input point cloud. This produces a simplified point cloud, which is not guaranteed to be a subset of the input. However, Ye et al. [YYJ22] argue that an actual subset of the input can be obtained by a matching algorithm, such as the nearest neighbor matching.

Because APSNet models the sampling process using an LSTM module, it can be trained to generate an arbitrary number of output samples. The LSTM sequentially produces one sample at a time while maintaining a hidden state that encodes the history of previously selected points. This design enables a single trained model to generate output point clouds of varying cardinality by adjusting the number of unrolled steps.

Wu et al. [WZPB23] contend that most point cloud samplers are based on generating new samples rather than selecting points directly using mathematical statistics. Inspired by the Canny edge detector [CAN87], the proposed APES network utilizes attention [VSP+17] and statistics to detect salient outline points (edge points). The authors note that the Canny edge detector selects edge pixels based on large intensity gradients in the neighborhood, which is equivalent to selecting pixels with large standard deviations of the intensities within the neighborhood. This observation allows us to generalize beyond pixel intensities to any per-pixel features computed between a center pixel and its neighborhood using a "measure of feature correlation" function. Wu et al. [WZPB23] transfer this idea to point cloud sampling by utilizing a k-nearest neighbor algorithm to obtain a local neighborhood for each 3D point. First, each 3D point is processed using multiple EdgeConv blocks [WSL+19] to obtain per-point feature vectors. Next, for each point, a local neighborhood is constructed by applying the k-nearest neighbor algorithm, from which a correlation map is computed as the attention [VSP+17] between the center point and its neighbors. This attention map represents the relative importance of the neighbors of a point. To assess how "edge-like" each point is, the standard deviation of its attention weights is calculated, and points with higher standard deviations are selected as a representative subset. This allows the network to output an arbitrary number of m sampled points by taking the points with m highest standard deviations. Similarly to previously discussed sampling approaches, APES is trained and evaluated using a pretrained and frozen task network.

The APES network is a task-aware sampling network, capable of selecting informative points from a point cloud rather than generating new ones. It supports flexible output sizes by ranking points based on their edge saliency and selecting the top m representatives. However, due to the network prioritizing boundary points, it might underrepresent interior regions of the shape, potentially missing important structures that are not located near edges. Additionally, the network is highly dependent on the hyperparameter k for the k-nearest neighbor sampling, where an overly small k might lead to context loss, while a large k might incorporate unrelated or distant points into the neighborhood.

Chen et al. [CCQ+26] argue that previous point cloud sampling strategies do not effectively capture complex relationships between neighbouring point clouds. The proposed GS-Net, based on graph neural networks [SGT+09], first converts each point from the input point cloud into a single graph node, and constructs an adjacency matrix A capturing k nearest neighbors of each point. After the graph is constructed, a structure encoder based on GraphSage layers [HYL17] is applied to aggregate local neighbourhood information and learn structural features for each node. The resulting multi-scale point-wise embeddings are aggregated to fuse local and global features into a single vector, and fed into an

Sampling Network	Task Network	Flexible output size
S-Net [DLA19] SampleNet [LMA20]	✓	✗
S-Net Progressive [DLA19] SampleNet Progressive [LMA20]	✓	✓
ASPNet [YYJ22]	✓	✓
Yang et al. [YSA23]	✗	✓
APES [WZPB23]	✓	✓
GS-Net [CCQ+26]	✓	✗

Table 2.1: Overview of influential sampling networks.

MLP to predict sampled point coordinates. Similarly to Dovrat et al [DLA19] or Lang et al. [LMA20], GS-Net is optimized end-to-end using a downstream task network. The optimization combines a task loss, based on the direct feedback from the downstream task network, and a proximity loss, which encourages the sampled points to remain close to the original point cloud. This approach effectively encodes the local relationships between points and captures multi-scale local details. However, constructing the graph for large point clouds can be computationally expensive and may introduce unwanted latency. Additionally, the authors note that GS-Net primarily focuses on single-object datasets, and extending it to multi-object or large-scale scenes may introduce additional challenges.

2.2.2 Sampler Summary

From the related work analyzed, we identified two core methodological differences between the samplers introduced in prior studies: the supervision paradigm employed during training, and the flexibility of the output size. The supervision paradigm refers to whether training involves task-specific supervision by directly minimizing the loss of a task network through gradient feedback, or by optimizing the sampler independently without access to the task loss during training. Output size flexibility captures whether a method generates subsets of a fixed size (and requires retraining for different subset sizes) or uses a single trained model to produce subsets of varying sizes. For all methods introduced above, Table 2.1 reports their characteristics in terms of the two overarching criteria. The sampling pipeline presented in this work is based on SampleNet [LMA20] and, consequently, employs a task-driven training paradigm while producing subsets of a fixed number of points.

2.3 Identified Gap

Our goal is to implement data-aware sampling of subsets from binary segmentations of 3D medical images, from which the original segmentation can be reconstructed with maximal accuracy. However, sampling voxel grids directly is challenging due to their rigid, predefined structure, high memory requirements [SEJ18, BTZ+21], and the difficulty of designing a sampling architecture that can generalize across different resolutions and voxel spacings [GLB+19]. For this reason, we propose to convert voxel grids containing binary segmentations into the point cloud domain, where we perform the sampling, and map the sampled points back to volumetric representation and extract the corresponding volumetric patches for downstream processing. The point cloud domain allows us to simplify the input point clouds to obtain a lightweight representation of the underlying object, removing redundant or similar points while preserving overall structure. It also requires less memory because only occupied points are stored.

All the previously discussed point cloud-based samplers achieve remarkable performance on benchmark datasets and demonstrate the effectiveness of task-aware point selection. By learning to identify relevant points, they reduce computational cost and improve performance across tasks such as classification, reconstruction, and segmentation. However, these methods have been evaluated on point clouds representing surface geometry, typically hollow or sparsely sampled 3D shapes, and their effectiveness on densely filled data remains unexplored. In addition, the samplers are trained and evaluated on clean, synthetic datasets, with testing on real-world data missing. As a result, the ability of these methods to capture real-world data and their robustness to artifacts, such as noise or outliers, remains unverified. Finally, supervised point cloud sampling networks, such as the work presented by Dovrat et al. [DLA19] or Lang et al. [LMA20] are trained using downstream task networks that operate on point clouds, using task losses computed on point cloud outputs. In contrast, our goal is to utilize a volumetric reconstruction network with a loss defined on 3D volumes, which requires adapting the training of the sampling pipeline to accommodate the volumetric-based loss.

This work addresses all of the aforementioned limitations. The proposed sampling pipeline is designed with the goal of optimizing sampling for the reconstruction of volumetric shapes. To reduce memory requirements and address the challenge of designing a sampler that generalizes to different input resolutions, we transform the sampling problem into the point cloud domain. We adopt a point cloud sampling architecture SampleNet [LMA20] and modify its loss to support volumetric data sampling and allowing processing of dense point clouds. Finally, we train and evaluate the proposed pipeline on two real-world datasets of 3D medical images, testing its robustness to realistic variability, noise, and artifacts.

CHAPTER 3

Methodology

We begin this chapter with the Problem Formulation in Section 3.1, defining the objects and notations used, and introducing a formal definition of the sampling problem. Additionally, we discuss the motivation behind the design choices of the proposed pipeline. The pipeline is divided into three components: *Voxel Grid Conversion*, *Point Cloud Sampling*, and *Patch Extraction*, each illustrated in Figure 3.1. The Voxel Grid Conversion takes the input voxel grid, representing the binary segmentation of a medical structure, and converts it to a point cloud representation, and is described in Section 3.2. Subsequently, the Point Cloud Sampling processes the point cloud and outputs a set of points representing the centers of the proposed volumetric patches. The underlying architecture for generating the patch proposals is described in more detail in Section 3.3. Next, the Patch Extraction component performs the inverse mapping from the point cloud domain to voxels, yielding a set of volumetric patches with their respective coordinates. Section 3.4 provides further details on how we implement the inverse mapping in a way that maintains full differentiability with respect to the generated coordinates from the previous step. Finally, in the last step, we process selected volumetric patches using a pretrained and fixed reconstruction network to obtain the complete shape of the input binary segmentation. We discuss the utilized volumetric reconstruction network in Section 3.5.

3.1 Problem Formulation

First, in Section 3.1.1 we provide definitions of objects and notations used in this Chapter. Section 3.1.2 then presents a formal definition of the problem of sampling volumetric patches. Finally, Section 3.1.3 discusses the motivation behind the design choices of the proposed pipeline.

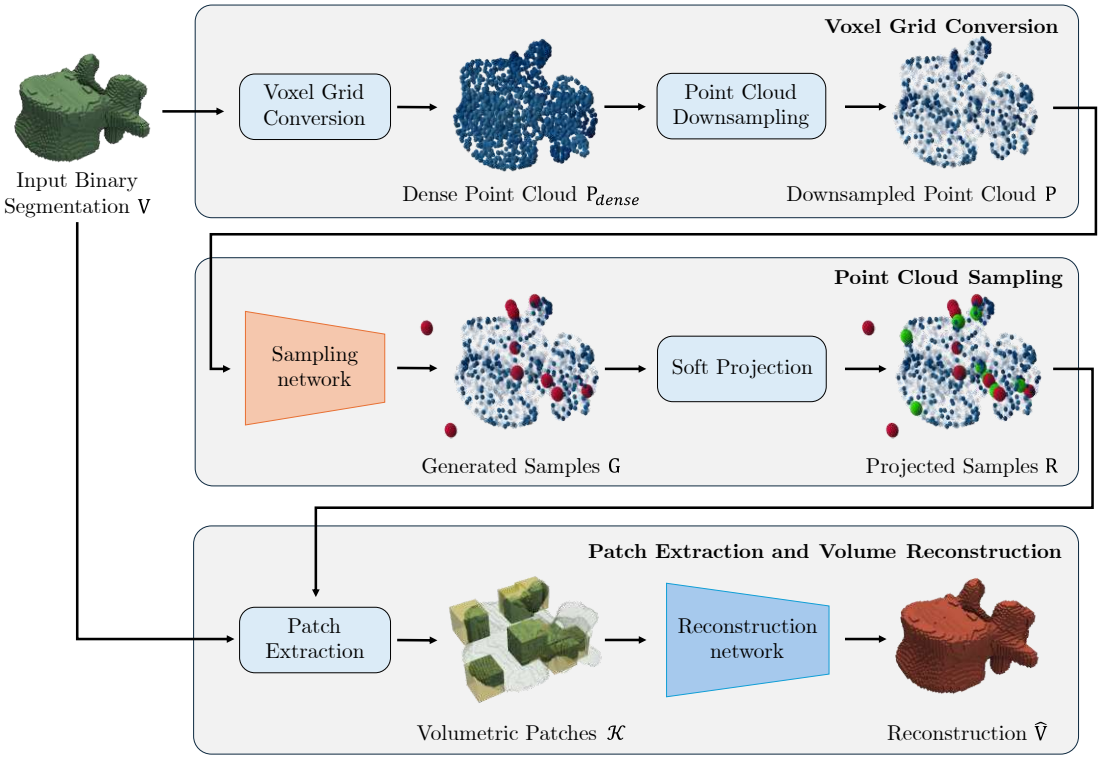


Figure 3.1: Overview of the proposed pipeline. First, we utilize two proposed modules to convert the input segmentation V to the point cloud domain and downsample it (Section 3.2). Next, we employ a sampling architecture by Lang et al. [LMA20] to process each point cloud and generate a simplified point cloud G. We adapt their Projection module to project the simplified point cloud G onto the input point cloud, producing a set of patch center proposals R, with modifications to its training and inference (Section 3.3). A proposed Extraction module then aligns extracted point samples R with input segmentation V and produces a set of volumetric patches (Section 3.4). Finally, we employ a reconstruction network by Amiranashvili et al [ALL+22], which processes the extracted patches and return the reconstruction $\hat{V}$ (Section 3.5).

3.1.1 Objects and Notations

Let $M_{X,Y,Z}(\mathcal{D})$ denote the set of all 3D matrices (voxel grids) of size $X \times Y \times Z$ with entries in the domain $\mathcal{D}$. Given a binary segmentation V of a 3D medical image, we have $V \in M_{W,H,D}(\{0,1\})$, where $W, H, D \in \mathbb{Z}$ are the spatial dimensions of the voxel grid. As an example, consider the result of a CT scan with a corresponding binary annotation obtained through expert reviewers [TTKORS20].

In close relation we define $\mathcal{P}_n$ as the set of all point clouds with n points in $\mathbb{R}^3$, where each point cloud is of the form $\{\mathbf{p_i}\}_{i=1}^n$ with $\mathbf{p_i} = (x_i, y_i, z_i)$ with $x_i, y_i, z_i \in \mathbb{R}$. We denote $\phi_\mu(P)$ as a *point cloud transformation function*, which maps a point cloud $P \in \mathcal{P}_n$ to a point cloud $\tilde{P} \in \mathcal{P}_n$, i.e. $\phi_\mu(P) : \mathcal{P}_n \to \mathcal{P}_n$, where μ denotes a set of statistics (e.g. minimal and maximal coordinates, or mean and standard deviation). Throughout this work, we assume ϕ_μ to be invertible and denote $\phi_\mu^{-1}(\tilde{P})$ as its inverse function.

We define a *cubic neighborhood* $C_s^{\mathbf{q}} \subset \mathbb{R}^3$ of edge length s, centered around point $\mathbf{q} \in \mathbb{R}^3$ as an axis aligned cube of edge length $s \in \mathbb{Z}, s \geq 1$ with $\mathbf{q} = (q_1, q_2, q_3)$ as the center point, i.e.,

$$C_s^{\mathbf{q}} := \{\mathbf{a} \in \mathbb{R}^3 \mid \max_i |a_i - q_i| \leq \frac{s}{2}\}, \; i = \{1,2,3\}. \tag{3.1}$$

Given a point $\mathbf{q} \in \mathbb{R}^3$, we define a *patch coordinate grid*

$$X^{\mathbf{q}} = \{\mathbf{x}_{i,j,k}\}_{i,j,k=1}^s \tag{3.2}$$

as the set of center points of a regular $s \times s \times s$ sampling grid over a cubic neighborhood $C_s^{\mathbf{q}}$, where

$$\mathbf{x}_{i,j,k} = \mathbf{q} + \begin{pmatrix} i - \frac{s+1}{2} \\ j - \frac{s+1}{2} \\ k - \frac{s+1}{2} \end{pmatrix}, \quad i,j,k = 1,\ldots,s. \tag{3.3}$$

Correspondingly, we define *patch labels* $Y^{\mathbf{q}}$ as a set of binary labels for each $\mathbf{x} \in X^{\mathbf{q}}$:

$$\mathbf{Y}^{\mathbf{q}} = \{y_{i,j,k}\}_{i,j,k=1}^s, \quad y_{i,j,k} \in \{0,1\}, \quad i,j,k = 1,\ldots,s.$$

Finally, let $P \in \mathcal{P}_n$. We define a *volumetric patch* centered at point $\mathbf{q}$ as $Q^{\mathbf{q}} = (X^{\mathbf{q}}, Y^{\mathbf{q}})$. We further define the *set of volumetric patches* associated with point cloud P as $K^P = \{Q^{\mathbf{q}} \mid \mathbf{q} \in P\}$. Let $\mathcal{K}$ denote the set of all volumetric patch sets, then we define $\mathcal{K}_n = \{K \in \mathcal{K}, |K| = n\}$ set of all volumetric patch sets containing n patches.

For convenience, we further define an indexing of a patch $Q = (X, Y)$ by a linear index $n \in \{1,\ldots,s^3\}$ via indices $i,j,k \in \{0,...,s-1\}$ where $n = i + s^2 \cdot j + s \cdot k + 1$ as:

$$X = \{\mathbf{x}_n\}_{n=1}^{s^3}, \quad \mathbf{x}_n = \mathbf{x}_{i,j,k},$$
$$Y = \{\mathbf{y}_n\}_{n=1}^{s^3}, \quad \mathbf{y}_n = \mathbf{y}_{i,j,k} \tag{3.4}$$

In the context of this work we assume there is a given *reconstruction model* denoted f_{rec}, which processes a set of n volumetric patches K_n and outputs a voxel grid $\hat{V} \in M_{W,H,D}(\{0,1\})$. We denote f_{samp} for a point cloud *sampler* as a function that projects a point cloud $P \in \mathcal{P}_n$ to a subset G of $m = |G|$ elements, i.e., $f_{\text{samp}} : \mathcal{P}_n \to \mathcal{P}_m$.

Finally, we denote $\|\cdot\|_2$ as the Euclidean norm of a vector $\mathbf{v}$, i.e.,

$$\|\mathbf{v}\| := \sqrt{\sum_{i=1}^{n} v_i^2}, \quad \mathbf{v} \in \mathbb{R}^n.$$

3.1.2 Problem Definition

In this thesis, we assume that we are given a voxel grid V representing a binary segmentation of a medical object. The objective is to select a representative set of m volumetric patches $K^* \in \mathcal{K}_m$ from V such that the reconstructed volume $\hat{V}$, obtained as $\hat{V} = f_{\text{rec}}(K^*)$, maximizes a reconstruction quality measure $\rho : M_{W,H,D}(\{0,1\}) \times M_{W,H,D}(\{0,1\}) \to \mathbb{R}$. Formally, this can be written as:

$$K^* = \underset{K \in \mathcal{K}_m}{\arg\max}\, \rho(V, \hat{V}), \quad \hat{V} = f_{\text{rec}}(K).$$

3.1.3 Problem Decomposition

Voxel Grid Conversion In this work, the input data are represented as voxel grids with a fixed resolution. However, the voxel grid structure makes it difficult to design a sampling architecture that generalizes well across different resolutions and voxel spacings [GLB+19]. Additionally, voxel grids explicitly store values for all voxels, including those representing empty space, resulting in significant memory requirements [BTZ+21]. We therefore introduce a simplification step that reduces the number of points while preserving the overall shape, yielding a lightweight approximation that captures the global geometry. This simplified point cloud is then used as a guide for subsequent point cloud sampling. For this reason, we design a *Voxel Grid Conversion* component (Section 3.2), which transforms the input voxel grid V to a point cloud P of n points.

Point Cloud Sampling Next, our target is to process P and obtain a set of m points R, which we can interpret as a set of centers of patches of a specified size s. By placing a cubic neighborhood in continuous space around each center point $\mathbf{r}_i \in R$, we define the corresponding set of volumetric patches $K^R \in \mathcal{K}_m$. To obtain R, we adapt a state-of-the-art sampler f_{samp} from Lang et al. [LMA20], chosen for its lightweight design and its ability to select points guided by the downstream task instead of using generic sampling

strategies. The sampler processes an input point cloud P and generates a new point cloud $G \in \mathcal{P}_m$, representing a subset that characterizes the underlying input point cloud P. However, because the architecture of f_{gen} does not restrict its output in any way, points of G are not guaranteed to be generated in the proximity of the input point cloud, and interpreting them as centers of volumetric patches could result in extracting patches without any content. To resolve this issue, we adapt the soft-projection operator from the same work, which provides a differentiable mechanism to move points close to the input point cloud P. This operator projects points $\mathbf{g}_i$ to $\mathbf{r}_i$, by representing them as weighted sums of their k nearest neighbors in P. This way, a volumetric patch centered around $\mathbf{r}_i$ is guaranteed to have an intersection with the underlying voxel grid (Section 3.3).

Patch Extraction In order to obtain the set of volumetric patches K^R and to provide the reconstruction network with the required input, we need to convert the sampled point cloud R into volumetric patches. For this purpose, we employ a Patch Extraction Module f_{extract}, which, for each sampled point $r \in R$, gathers the neighborhood voxel coordinates and extracts their corresponding binary labels from the underlying voxel grid (Section 3.4).

Reconstruction Finally, after we have extracted the set of volumetric patches K^*, we apply the reconstruction operation f_{rec} to produce the reconstruction $\hat{V}$ (Section 3.5). This step is necessary for training of the sampling component, which integrates the subsequent task performance into its training. Additionally, since our goal is to maximize the reconstruction performance measure ρ, we use $\hat{V}$ to evaluate the effectiveness of the sampling component.

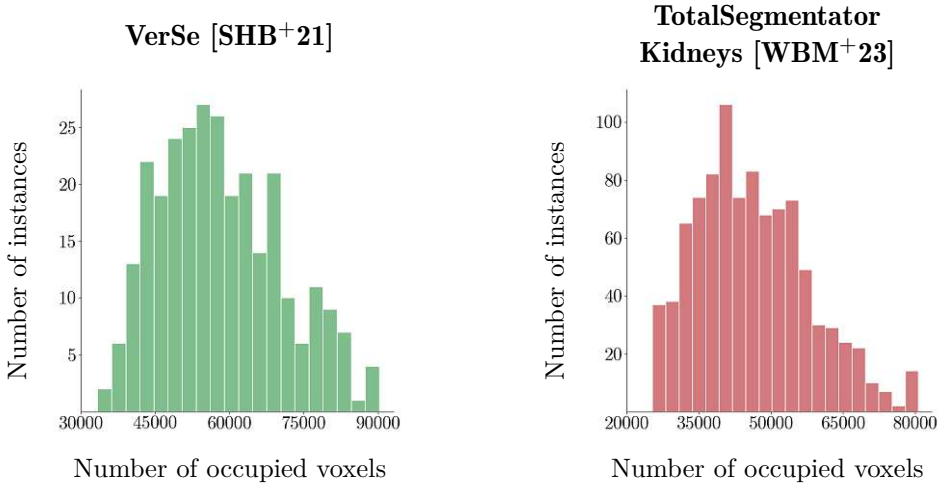
3.2 From Voxel Grids to Point Clouds

In our pipeline, we propose performing the core sampling process in the point-cloud domain. Compared to the original voxel grid structure, point clouds require less memory, as only occupied voxels are considered as points [LTLH19]. Moreover, the point cloud representation allows for generating points in a smooth manner, without any discretization steps [LTLH19]. Lastly, point clouds are not limited to a specific spatial resolution, which simplifies training across multiple resolutions of medical scans [LTLH19].

The purpose of this component is to process an input voxel grid V into a point cloud $P \in \mathcal{P}_n$, which is performed using two steps.

First, we convert V into $P_{\text{dense}} \in \mathcal{P}_l$, where l is a number of occupied voxels in V. Each occupied voxel is transformed into a point in 3D space with the indices in the volume as its coordinates. Hence, P_{dense} is described by:

$$P_{\text{dense}} = \{(i,j,k) \in \mathbb{R}^3 \mid 0 \leq i < W,\ 0 \leq j < H,\ 0 \leq k < D,\ V[i,j,k] = 1\}, \tag{3.5}$$



(a) Grid resolution $128 \times 120 \times 90$.

(b) Grid resolution $85 \times 85 \times 110$.

Figure 3.2: Distributions of the number of occupied voxels in each sample across selected datasets with varying grid resolution.

However, such conversion produces point clouds of varying sizes, depending on the number of occupied voxels in the respective segmented scans. For reference, Figure 3.2 presents histograms showing the distribution of occupied voxel counts across the datasets used for evaluation, which are described in more detail in Section 4.2.

To ensure comparability with existing state-of-the-art approaches, which often operate on point clouds of a constant size [DLA19, LMA20, YSA23], we downsample each dense point cloud P_{dense} obtained from the corresponding voxel grid V, to a fixed number of points. This produces a new point P_{simpl} with a consistent size n. Additionally, the downsampling step addresses a practical challenge in our pipeline: as each converted point cloud contains tens of thousands of points (see Figure 3.2), direct processing could overwhelm the sampling architecture by providing too many similar points [LLZ21]. By reducing the number of points, we create a sparse yet representative object encoding that preserves the overall shape while simplifying the input for the sampler. At the same time, we retain the original full-resolution voxel grid, which allows us to extract volumetric patches at full resolution around each generated patch center provided by the sampler. This reduction in the number of points also reduces memory requirements, speeds up training of the sampling network, and results in shorter overall inference times.

As we want to simplify the point cloud in a way that uniformly spreads out the points, we utilize a Sobol sequence [Sob76], which generates quasi-random points with uniform and distributed coverage of space. The Sobol sequence consists of points within a d-dimensional hypercube $[0,1]^d$, from which we draw n samples in the three-dimensional domain $[0,1]^3$. We then normalize the dense point cloud to range $[0,1]$ to match the scale of Sobol samples, and finally obtain the simplified point cloud P_{simpl} by nearest neighbor

matching [CH67] between the Sobol points and the normalized point cloud. Formally,

$$P_{\text{simpl}} = \{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_n\},\ \tilde{\mathbf{p}}_k = \arg\min_{p \in \tilde{P}_{\text{dense}}} \|\mathbf{s}_k - \mathbf{p}\|_2,\ k = 1, \ldots, n,$$

where $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\} \subset [0,1]^3$ is a Sobol sequence.

Finally, we apply a point cloud transformation operation ϕ_μ to normalize the point cloud P and save the normalization statistics μ for later use:

$$P = \phi_\mu(P_{\text{simpl}})$$

3.3 Point Cloud Sampling

Once the voxel grid is converted into a point cloud $P \in \mathcal{P}_n$, this point cloud is processed using a Point Cloud Sampling component, as depicted in Figure 3.1. The purpose of Point Cloud Sampling is to retrieve a set of points that provide a compact representation of the input point cloud, based on how informative each point is for the reconstruction task. Formally, the Point Cloud Sampling component is defined as a function

$$f_{\text{samp}} : \mathcal{P}_n \to \mathcal{P}_m, \quad f_{\text{samp}}(P) = R,$$

which outputs a set of sampled points R used as centers of volumetric patches. For a given set of sampled points R, the corresponding volumetric patches K^R are uniquely determined as

$$K^R = \{Q^{\mathbf{r}} \mid \forall \mathbf{r} \in R\}.$$

In accordance with Section 3.1.2, our goal is to select a representative patch set $K^* \in \mathcal{K}_m$ that maximizes reconstruction measure ρ. We achieve this by optimizing over the candidate point cloud $R \in \mathcal{P}_m$ and then extracting the associated patch set:

$$R^* = \arg\max_{R \in \mathcal{P}_m} \rho(V, f_{\text{rec}}(K^R)), \quad K^* = K^{R^*}.$$

In our pipeline, f_{samp} is implemented as two stages: the *Samples Proposal stage* f_{gen} and the *Projection stage* f_{proj}, illustrated in more detail in Figure 3.3. The Sample Proposal stage produces a preliminary set of sampled points, while the Projection stage refines their locations with respect to the underlying point cloud P to ensure that, when interpreted as patch centers, each point defines a volumetric patch that is in close proximity or intersects the underlying voxel. Formally, given an input point cloud of 3D coordinates P, the two stages operate sequentially as:

$$R = f_{\text{proj}}(f_{\text{gen}}(P), P), \tag{3.6}$$

yielding the set of proposed patch centers $R \in \mathcal{P}_m$.

In this work, we implement f_{gen} as a neural network, mapping the input point cloud P to a set of m sampled points, $G \in \mathcal{P}_m$. The architecture of f_{gen} follows Lang et al. [LMA20] and employs a linear output layer without normalization or activation functions. For this reason, the point coordinates are generated unconstrained in the 3D space. However, it is possible for samples generated freely in the continuous space to be positioned far from the input, and as a result, each volumetric patch extracted around such a sample will not contain any meaningful part of the underlying structure. For this reason, the f_{proj} is introduced: a projection mechanism that differentiably maps all generated samples from G onto local point neighborhoods in P, illustrated in Figure 3.6. Because projected samples are represented as convex combinations of points from P (cf. Equation 3.3), patches extracted using such projected samples are guaranteed to lie within the convex hull of P and V. In practice, this projection significantly reduces the likelihood of sampling patches far from the underlying structure by constraining projected samples to lie within the convex hull of the shape. Both the sample Proposal architecture and the soft projection module are adopted from the framework proposed by Lang et al. [LMA20]. However, we adapt the projection module to better suit our proposed pipeline and goal. Unlike Lang et al. [LMA20], whose method operates on full-resolution point clouds with a fixed number of points and aims to select an actual subset of the input points, in our settings, the input point cloud represents a simplified geometry of a volumetric shape. Consequently, we do not constrain the sampler to select a discrete subset of input points and instead allow it to propose arbitrary locations in continuous space around the simplified geometry. This enables exploration of intermediate regions that are not explicitly represented in the downsampled point cloud.

Samples Proposal Stage The Samples Proposal Stage f_{gen} is formalized as a function

$$f_{\text{gen}} : \mathcal{P}_n \to \mathcal{P}_m, \qquad f_{\text{gen}}(P) = G,$$

which maps the input point cloud P to a generated point cloud G consisting of m points. Each point in G is unconstrained in the 3D space, as the network employs a linear output layer without normalization or activation.

The architecture of f_{gen} follows SampleNet [LMA20]. First, a series of 1×1 Convolutional layers, interleaved with ReLU activation and Batch Normalization, processes each point in the input point cloud independently, resulting in per-point feature vectors. A symmetric max-pooling operation then aggregates these features into a single global feature vector, which ensures that the architecture is invariant to the ordering of points in P. Finally, a linear layer followed by Batch Normalization and ReLU, and a final linear layer produce the three-dimensional coordinates of each generated point in G.

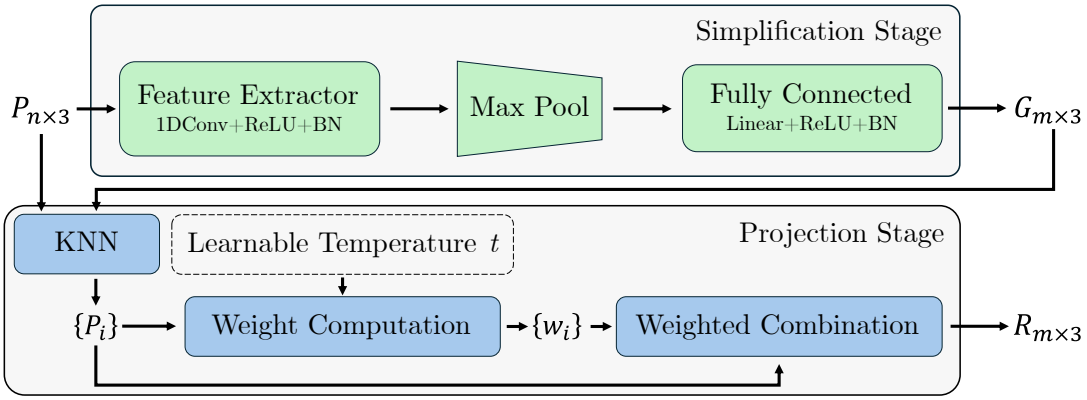


Figure 3.3: Point Cloud Sampling component consists of two stages: the Simplification stage employs a neural architecture processing an input of n points $P \in \mathcal{P}_m$ into a set of generated points $G \in \mathcal{P}_m$, freely positioned in the continuous space. To obtain samples closer to the input point cloud, a Projection stage applies a soft projection operator that maps generated samples onto a local neighborhood defined by the underlying point cloud. Image adapted from Lang et al. [LMA20].

Projection Stage The purpose of the Projection stage is to ensure that each generated point serves as a meaningful patch center, such that a volumetric patch placed around it is in close proximity or ideally intersects the underlying voxel grid rather than sampling empty space. Since the Proposal stage produces points freely in the continuous space without explicit spatial constraints, some of them may lie far from the underlying object. To address this issue, in this stage, the points are projected onto the input point cloud, ensuring that all sampled locations correspond to meaningful regions of the data. This projection is implemented by adapting the Soft-projection operator proposed by Lang et al. [LMA20], illustrated in Figure 2.12.

Formally, given point clouds $G \in \mathcal{P}_m$ and $P \in \mathcal{P}_n$, both embedded in $\mathbb{R}^3$, the Projection Stage f_{proj} maps G onto the geometric structure represented by P, producing a point cloud $R \in \mathcal{P}_m$:

$$f_{\text{proj}} : \mathcal{P}_m \times \mathcal{P}_n \to \mathcal{P}_m, \qquad f_{\text{proj}}(G, P) = R,$$

In this stage, for each generated point $\mathbf{g} \in G$, $\mathcal{I}_P^{\mathbf{g}}$ denotes the index set of its k-nearest neighbors in P. Each projected point $\mathbf{r} \in R$ is then computed as a weighted sum of its respective neighbors:

$$\mathbf{r}_i = \sum_{j \in \mathcal{I}_P^{\mathbf{g}_i}} w_j \, \mathbf{p}_j, \quad w_j = \frac{\exp(-\|\mathbf{g}_i - \mathbf{p}_j\|_2^2 / t^2)}{\sum_{k \in \mathcal{I}_P^{\mathbf{g}_i}} \exp(-\|\mathbf{g}_i - \mathbf{p}_k\|_2^2 / t^2)}. \tag{3.7}$$

Here, a learnable temperature parameter t controls the sharpness of the weighting and is optimized jointly with the weights of the sampling network during training. The

neighborhood size $k = |\mathcal{I}_{\mathrm{P}}^{\mathbf{g}}|$ plays a role in the choice of sampled points, where a small neighborhood size demotes point cloud exploration, while an excessive size may result in loss of local context [LMA20]. In practice, k is treated as a hyperparameter and chosen empirically based on validation [LMA20].

3.4 Extraction of Volumetric Patches

The sampling component f_{samp} produces a set $R \in \mathcal{P}_m$ of points, each projected onto its respective local region on the underlying input point cloud. We treat each projected sample r as a proposal for the center of a patch around it. The goal of the Patch Extraction Module f_{extract} is to convert each projected point $\mathbf{r} \in R$ into a volumetric patch of edge length s, producing the set of m patches $K^R \in \mathcal{K}_m$ extracted from the voxel grid V. Formally:

$$f_{\text{extract}} : \mathcal{P}_m \times \mathbb{R} \times M_{W,H,D}(\{0,1\}) \to \mathcal{K}_m, \quad f_{\text{extract}}(R, K, V) = K^R$$

The projected samples $\mathbf{r} \in R$ are expressed as convex combinations of points from the input point cloud P and do not form a subset of the input point cloud. For this reason, it is not possible to directly voxel-align the projections with the voxel grid containing the binary shape segmentation, and a method is required that can extract volumetric patches around continuous coordinates rather than using discrete voxel indices.

Accordingly, the employed Patch Extraction module converts the proposed patch centers $\mathbf{r} \in R$ into a set of volumetric patches K^R, suitable for the reconstruction task, and converts the data representation back to a voxel grid, as illustrated in Figure 3.4. This extraction process must be fully differentiable with respect to the outputs of the preceding stage (i.e., the coordinates in point cloud R), since the sampler is trained using gradients propagated from the reconstruction network, which flow through the extraction operation. Furthermore, the reconstruction network, outlined in Section 3.5, performs inference-time optimization, using binary labels for each patch coordinate as ground truth. For this reason, it is necessary to extract not only the patch coordinates but also their corresponding binary labels from the underlying input voxel grid.

Let V denote an input voxel grid, and let $\mathbf{r} = (r_x, r_y, r_z) \in R$ be a projected point obtained from the sampling stage. We first map $\mathbf{r}$ back to the voxel grid coordinate domain using the inverse transformation ϕ_μ^{-1}, where μ are the transformation statistics from Section 3.2 (Equation 3.2). We denote the resulting point as

$$\tilde{\mathbf{r}} = (\tilde{r}_x, \tilde{r}_y, \tilde{r}_z) = \phi_\mu^{-1}(\mathbf{r}).$$

Finally, let s denote the edge length of a cubic neighborhood $C_s^{\mathbf{r}}$. The coordinates of the patch and the corresponding voxel labels are then obtained as

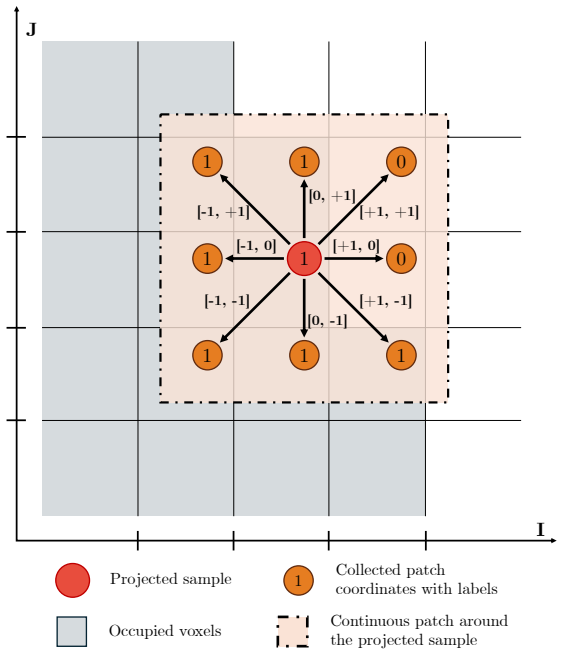


Figure 3.4: Patch Extraction around a projected point. A 3D projected point (red) is positioned within the voxel grid, and a fixed set of relative offsets is applied to collect neighboring coordinates (orange). Each coordinate is assigned a binary occupancy value from the underlying voxel grid using the nearest-neighbor matching.

$$
\begin{aligned}
X^{\mathbf{r}} &= \{(\tilde{r}_x + i, \tilde{r}_y + j, \tilde{r}_z + k) \mid i, j, k \in \{\lfloor -s/2 \rfloor, \ldots, \lfloor s/2 \rfloor\}\}, \\
Y^{\mathbf{r}} &= \{V[\mathrm{round}(\mathbf{x})] \mid \forall \mathbf{x} \in X^{\mathbf{r}}\},
\end{aligned}
\tag{3.8}
$$

where $\mathrm{round}\colon \mathbb{R}^3 \to \mathbb{Z}^3$ is the component-wise rounding operator defined by

$$\mathrm{round}(\mathbf{x}) = (\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \lfloor x_3 \rfloor), \ \mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3.$$

Finally, we extract the volumetric patches corresponding to the sampled points R, obtaining the set K^R. We treat each projected point as the center of a continuous patch and extract patch coordinates using relative offsets from the respective patch center. This allows us to obtain the set of patch coordinates in a differentiable manner.

3.5 Reconstruction from Volumetric Patches

In the final step, f_{rec} processes extracted volumetric patches K^R associated with R, and reconstructs a complete shape $\hat{V}$ in the same resolution as the input voxel grid V.

In the proposed pipeline, we incorporate the state-of-the-art architecture of Amiranashvili et al. [ALL+22], as it successfully targets the reconstruction of segmentations of medical

shapes. More details on this architecture are provided in Section 2.1.2. The architecture takes a single 3D coordinate and a d-dimensional latent vector, which encodes a shape-specific representation, and predicts an occupancy probability at that coordinate. We pretrain the reconstruction network separately from the sampler and freeze its parameters. The shape reconstruction then consists of two stages: *Shape-specific Encoding* and *Reconstruction Inference*, both illustrated in Figure 3.5.

3.5.1 Shape-specific Encoding

The reconstruction procedure starts by obtaining a shape-specific encoding of the target shape. Given a set of sampled patches K^R and a randomly initialized d-dimensional latent vector $z \sim \mathcal{N}(0, 10^{-4}) \in \mathbb{R}^d$, the goal is to obtain a latent vector $z_{\text{rec}} \in \mathbb{R}^d$, representing the specific encoding of the shape being reconstructed.

For each set of patch coordinates X^i and the corresponding ground truth occupancy labels Y^i, f_{rec} predicts occupancy values for each patch coordinate:

$$\hat{Y}^i_j = f_{\text{rec}}(X^i_j, z), \quad j \in \{1, 2, ...s^3\} \tag{3.9}$$

where X^i_j is the j-th coordinate of the i-th patch, $\hat{Y}^i_j \in [0, 1]$ is the predicted occupancy, and s is the edge length of each patch. The shape-specific latent vector z_{rec} is then obtained by minimizing the reconstruction loss over all patch coordinates:

$$z_{rec} = \arg\min_z \sum_{i=1}^{m} \sum_{j=1}^{s^3} \mathcal{L}_r(\hat{Y}^i_j, Y^i_j), \tag{3.10}$$

where L_r is a reconstruction loss, comparing predicted occupancy $\hat{Y}^i_j$ to the ground truth Y^i_j. During this optimization, the latent vector z is updated through a fixed number of gradient descent steps (see black rectangle in Figure 3.5). This process encodes the specific shape into z_{rec}, which is then used in the following step to obtain the full-resolution reconstruction.

3.5.2 Reconstruction Inference

After the latent vector optimization, the function $f_{\text{rec}}(\cdot, z_{\text{rec}})$ encodes the reconstructed shape as a continuous implicit function, which can be sampled at arbitrary points to obtain the shape at any desired level of detail. As we aim to reconstruct a shape $\hat{V}$ with the same resolution as the input, we utilize a coordinate grid Ω that corresponds to the centers of all voxels in a $W \times H \times D$ lattice. Then the full shape may be recovered through $\hat{V} = f_{\text{rec}}(\Omega, z_{\text{rec}})$, where

$$\Omega = (\{0, 1, \ldots, W-1\} \times \{0, 1, \ldots, H-1\} \times \{0, 1, \ldots, D-1\}) + \frac{1}{2}(s_x, s_y, s_z). \tag{3.11}$$

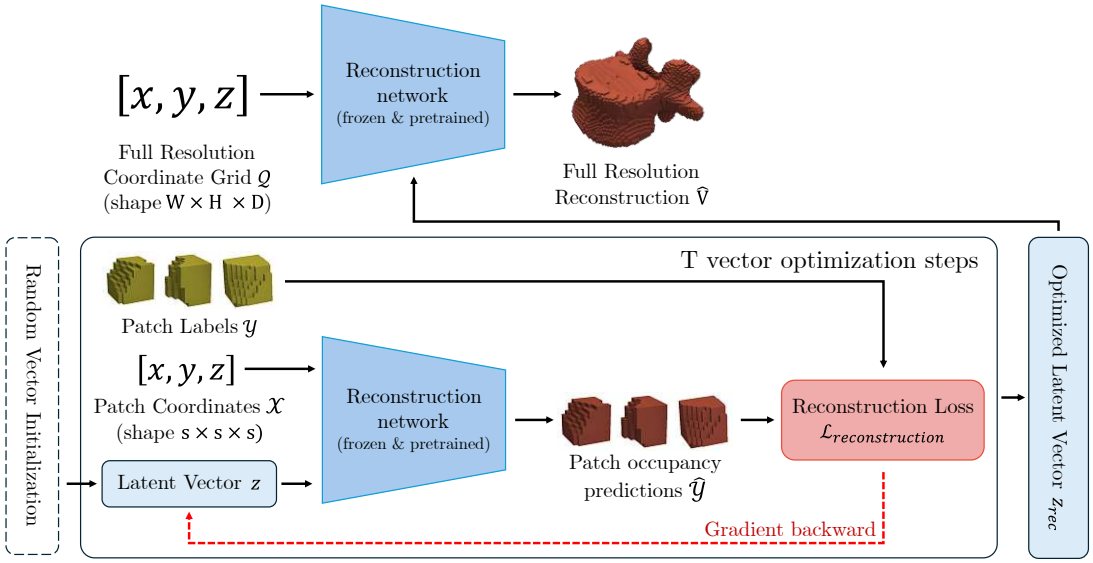


Figure 3.5: Reconstruction of a full resolution segmentation from sparse patches. A randomly initialized latent vector z is optimized T times by predicting the occupancy values $\hat{Y}$ for the coordinates of all sampled patches X. Afterwards, the optimized latent vector, z_{rec}, is used in one more pass where all discrete coordinates creating the full resolution coordinate grid Ω are evaluated.

Here, W, H, D denote the dimensions of a full resolution grid, corresponding to the dimensions of the input voxel grid V, and (s_x, s_y, s_z) denote the voxel sizes along each axis. See the top row of Figure 3.5 for illustration.

3.6 Model Training

To optimize the sampling architecture, we follow Lang et al. [LMA20] and utilize (i) a sampling loss term $\mathcal{L}_s$ and (ii) a reconstruction loss term $\mathcal{L}_r$, each computed from a different stage of the sampling pipeline, as illustrated in Figure 3.7. The overall loss $\mathcal{L}_{\text{total}}$ is then computed as a weighted sum of the two loss terms as:

$$\mathcal{L}_{\text{total}} = \lambda_s \cdot \mathcal{L}_s + \lambda_r \cdot \mathcal{L}_r. \tag{3.12}$$

The sampling loss provides general guidance to the network based solely on the positions of generated and projected points, while the reconstruction loss provides direct feedback based on the reconstruction performance.

Sampling Loss The sampling loss $\mathcal{L}_s$ provides general geometric guidance to the sampling network by encouraging a well-distributed and input-aware set of points, as well as promoting exploration of the entire object. It is computed based on the input point cloud P, the generated point cloud G, and the projected point cloud R, as illustrated in Figure 3.7. We compute the loss as the sum of two terms: the proximity loss and the repulsion loss.

The proximity loss $\mathcal{L}_{\text{proximity}}$ encourages points from G to be generated closer to the input point cloud P. Maintaining generated points near the surface of the input point cloud stabilizes the subsequent projection step. When a generated point lies far from the input surface, even small updates to its position during training can lead to large displacements of its projected counterpart, resulting in unstable learning. In comparison, generating a point close to the input results in precise projections with the ability to perform finer movements. The proximity loss is defined as:

$$\mathcal{L}_{\text{proximity}}(G, P) = \frac{1}{m} \sum_{i=0}^{m-1} \min_{0 \leq j < |P|} \|\mathbf{g}_i - \mathbf{p}_j\|_2^2, \quad \mathbf{g}_i, \mathbf{p}_j \in \mathbb{R}^3 \tag{3.13}$$

where $m = |G|$. Contrary to Lang et al. [LMA20], we do not employ a maximal nearest neighbor loss, as we found that it does not improve sampling accuracy or accelerate convergence.

The purpose of the repulsion loss $\mathcal{L}_{\text{repulsion}}$ is to prevent patch centers, and consequently the extracted patches, from collapsing onto each other by enforcing a repulsive interaction between points. It is defined as:

$$\mathcal{L}_{\text{repulsion}}(R) = \sum_{\{i,j\} \in \mathcal{I}_{\mathcal{R}}} \frac{1}{\|\mathbf{r}_i - \mathbf{r}_j\|_2^2}, \tag{3.14}$$

where $\mathcal{I}_{\mathcal{R}}$ is the set of indices for all possible pairs of points in R. Compared to Lang et al. [LMA20], we reformulate the repulsion loss to accommodate point clouds with interior points. The original formulation depends on distances between projected samples $\mathbf{r}$ and the input point cloud P, and implicitly assumes that the input points represent the outer surface. In contrast, our definition of $\mathcal{L}_{\text{repulsion}}$ does not depend on the spatial distribution of the input point cloud, but rather encourages a point spread based only on positions of the samples $\mathbf{r}$ themselves.

Finally, the overall sampling loss is defined as a weighted sum of the two loss terms presented above:

$$\mathcal{L}_s = \lambda_{\text{proximity}} \cdot \mathcal{L}_{\text{proximity}} + \lambda_{\text{repulsion}} \cdot \mathcal{L}_{\text{repulsion}}. \tag{3.15}$$

In the original loss formulation of Lang et al. [LMA20], the sampling loss $\mathcal{L}_s$ has an additional loss term, the projection loss. In order to accommodate the sampler into our

proposed volumetric pipeline, we do not include it in out overall loss. In the formulation of Lang et al. [LMA20], the projection loss is defined as the temperature parameter t^2 (cf. Section 3.3). The learnable temperature parameter t controls the distribution of the projection weights $\{w_i \mid i \in \mathcal{I}_P(g)\}$, and as $t \to 0$, the projection approximates the nearest-neighbor matching (Figure 3.6). In Lang et al. [LMA20], the aim is to sample an actual subset of the input point cloud, which is achieved by including the temperature parameter in the loss and encouraging it to approach zero as training progresses, resulting in a nearest-neighbor approximation. However, approximating nearest-neighbor matching and producing an actual subset of the input point cloud does not align with the design of our proposed pipeline. We treat the input point cloud as a simplified shape reference and generate samples in between the original points. Training the temperature parameter to approach zero and using approximate nearest-neighbor matching would constrain each generated point to lie exactly on a previously downsampled point cloud. This would restrict patch selection to only the locations corresponding to the downsampled points, ignoring potentially informative regions in between. In our implementation, we retain the learnable temperature parameter but do not enforce its convergence to zero through any loss, allowing the network to freely explore the intermediate regions. Additionally, during inference, Lang et al. [LMA20] replace the soft projection module with the k-nearest-neighbor matching, which selects actual points from the input point cloud. Similarly, to avoid restricting the projected points to the discrete locations of the downsampled input, we do not replace the nearest neighbor matching during inference and keep the soft projection module, including the temperature parameter.

Reconstruction Loss The reconstruction loss provides direct feedback based on the quality of the volumetric reconstruction. Following Amiranashvili et al. [ALL+22], we utilize the soft dice score and binary cross entropy (BCE) in a per-voxel manner, defined by:

$$\mathcal{L}_{\text{Dice}} = \frac{2\sum_{i=1}^{N} \hat{v}_i v_i + \epsilon}{\sum_{i=1}^{N} \hat{v}_i + \sum_{i=1}^{N} v_i + \epsilon} \tag{3.16}$$

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} \left[v_i \log(\hat{v}_i) + (1 - v_i)\log(1 - \hat{v}_i)\right], \tag{3.17}$$

where $\hat{v}_i$ denotes the predicted occupancy value from the linearized reconstructed voxel grid $\hat{V} \in M_{W,H,D}(\{0,1\})$, $v_i \in \{0,1\}$ is the ground truth occupancy value from the flattened input voxel grid $V \in M_{W,H,D}(\{0,1\})$, $N = W \cdot H \cdot D$ is the total number of voxels in V, and ϵ is a small positive constant to prevent division by zero.

Then the loss of the reconstruction network $\mathcal{L}_r$ is defined as the sum of the Dice loss and the binary cross entropy:

$$\mathcal{L}_r = \mathcal{L}_{\text{Dice}} + \mathcal{L}_{\text{BCE}}. \tag{3.18}$$

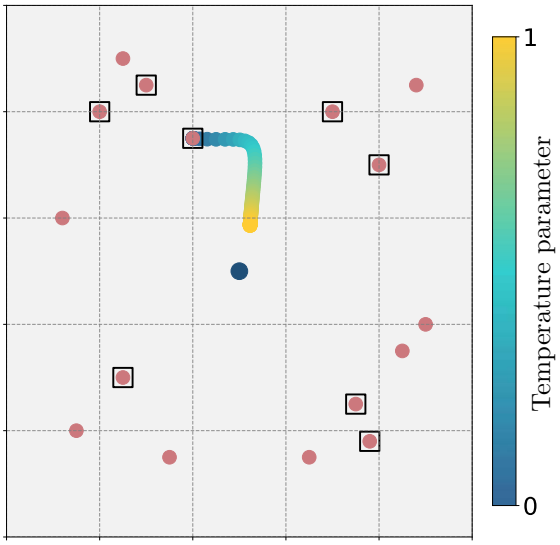


Figure 3.6: Soft projection operator with learnable temperature parameter t from the top view. Using an input point cloud as a reference (red), the generated point (dark blue) is expressed as a sum of its nearest neighbors (black squares), scaled by a temperature parameter. As the temperature approaches zero, the soft operation approximates nearest-neighbor matching.

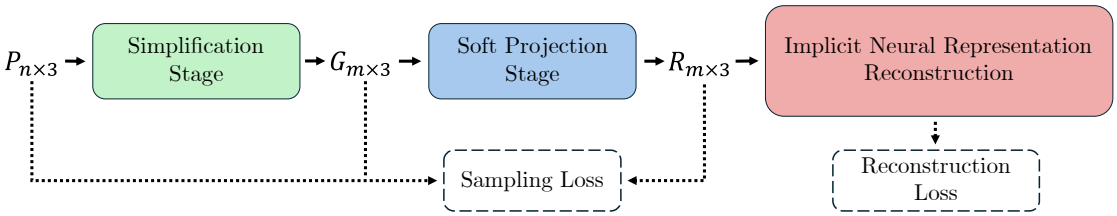


Figure 3.7: The sampling component is optimized using two loss terms, each being computed from a different stage.

The combination of both losses is a suitable choice for volumetric reconstruction: the Dice loss measures overall overlap between two segmentations, providing a global assessment of reconstruction quality, while the BCE loss evaluates each voxel independently, preserving local accuracy.

In summary, the proposed sampling pipeline converts binary segmentations represented as voxel grids into a point cloud domain. A sampling architecture, based on the work of Lang et al. [LMA20], processes the input point cloud and returns a smaller point cloud, representing the most informative points for the reconstruction task. We treat each sampled point as the center of a cubic neighborhood, split the neighborhood into a uniform sampling grid, and obtain coordinates for sampling along with their respective underlying binary labels. Extracting patch coordinates and binary labels for each sampled point

yields a set of volumetric patches, which are processed using a reconstruction network based on implicit neural representation, resulting in a reconstruction of the input voxel grid. The sampling architecture is trained using a combination of the sampling and reconstruction losses. The sampling loss encourages samples to be generated closer to the input and promotes sample spread, while the reconstruction loss assures generated samples are optimized for the reconstruction task. Compared to Lang et al. [LMA20], we modified both losses to better fit our pipeline: the sampling loss was adapted to handle point clouds containing points inside, and the reconstruction loss was modified to operate on volumetric outputs. We further removed the projection loss, since it encourages the sampler to select actual subsets of the input point cloud, which does not align with our goal of sampling freely around the input.

CHAPTER 4

Experimental Setup

This chapter outlines the experimental setup used to evaluate the proposed sampling approach. It establishes a framework for evaluation, ensuring consistency, comparability, and reproducibility across all experiments. Section 4.1 outlines the experiments designed to address our research questions. In Section 4.2 we outline the datasets used to evaluate our pipeline, including their preprocessing operations. Two baseline approaches are detailed in Section 4.3. Implementation details, including network architecture decisions and hyperparameter settings, are provided in Section 4.4. Finally, Section 4.5 describes the evaluation metrics and validation strategies employed to ensure robust and reliable results.

4.1 Experimental Pipeline

This section introduces the evaluation framework of the proposed sampling pipeline. We assess the performance of the sampling model by measuring its effect on reconstruction accuracy. We employ a pre-trained reconstruction network by Amiranashvili et al. [ALL+22]. The reconstruction network is kept fixed across all experiments, ensuring that observed differences in reconstruction quality can be attributed solely to the sampling strategy. In addition to the proposed method, the evaluation includes a comparison with two baseline sampling approaches, which are described in detail in Section 4.3.

The primary objective of the experimental evaluation is to analyze how the patch configuration (i.e. the number of patches and the patch size) affects reconstruction performance, since different configurations provide varying amounts of spatial context to the reconstruction network, which addresses RQ1 and RQ2. To this end, the proposed sampling pipeline is evaluated across multiple patch configurations, and reconstruction quality is compared using metrics described in Section 4.5.1. We consider configurations with $N = \{4, 8, 12\}$ patches, where each patch is a cubic region of $K \times K \times K$ voxels, with $K = \{9, 11, 13\}$, chosen to cover a wide range of sampling densities. These configurations

range from very sparse sampling with few small patches to denser coverage with more or larger patches. During our experiments, we observed that configurations with $N < 4$ and $K < 9$ provide insufficient spatial context to the reconstruction network, while configurations with $N > 12$ and $K > 13$ require substantially more memory and training time, making them impractical for our experiments.

Additionally, we investigate how the output of the patch sampler changes when it is trained using a reconstruction network of reduced accuracy. In practice, reconstruction networks may be less accurate due to noisy datasets, incomplete annotations, or other imperfections encountered in real-world applications [MC24]. In order to simulate different performance levels of the reconstruction model, we vary the number of optimization steps used to refine the shape-specific latent code z during reconstruction. Smaller values of T provide fewer updates to the latent vector, resulting in less shape-specific information and lower-quality reconstructions, whereas larger T values produce a more optimized latent vector and higher-quality reconstructions. By adjusting the number of latent optimization steps to $T = \{33, 66, 100\}$, we obtain reconstruction models of progressively increasing quality. We refer to the reconstruction setting with $T = 33$ as *low-quality*, $T = 66$ as *medium-quality*, and $T = 100$ as *high-quality*, which we consider the default reconstruction setting. The values of T were selected using an ablation study (Section 5.4.3), which showed that $T = 100$ provides a suitable trade-off between reconstruction quality and computational efficiency, as fewer optimization steps still yield comparable performance while reducing computational overhead and memory required for gradient tracking.

4.2 Datasets and Preprocessing

For our experiments, we select two publicly available datasets of 3D medical image segmentations from CT scans. From TotalSegmentator [WBM+23], which contains segmentations of multiple different medical shapes, we select kidneys, and then select VerSe [SHB+21], which contains segmentations of vertebrae. These datasets and shapes were selected based on the geometrical complexity of each shape and the voxel resolution of the segmentations. We describe both datasets in more detail in Sections 4.2.1 and 4.2.2 and summarize their properties in Table 4.1.

We filter samples in both datasets using dataset-specific criteria to remove invalid or corrupted entries. Afterwards, each sample is centered and cropped. The cropping dimensions are determined empirically by computing the largest 3D bounding box within dataset shapes and enlarging it by 10% in each dimension. It is worth noting that the crop does not affect the sampling process, as only occupied voxels are converted into the point cloud. The advantage of removing empty border regions is in the reduction of reconstruction time and memory requirements for the reconstruction network, as fewer coordinates need to be processed and stored during gradient computation. Similarly, the centering procedure, motivated by Amiranashvili et al. [ALL+22], does not influence sampling due to each point cloud being normalized in a later step.

Dataset	Shape	Modality	Voxel size [mm]	Resolution	Dataset size
TotalSegmentator [WBM+23]	Kidney	CT	1.5×1.5×1.5	85 × 85 × 110	300
VerSe [SHB+21]	Vertebra	CT	1.0×1.0×1.0	128×120×90	287

Table 4.1: Overview of benchmarking datasets.

4.2.1 TotalSegmentator Kidneys

TotalSegmentator dataset [WBM+23] is a collection of 1204 CT scans. It includes manual segmentations of 104 anatomical structures, covering a range of bones, organs, muscles, and vessels, each manually segmented by an experienced physician. To ensure a consistent representation across the dataset, all scans were resampled to an isotropic resolution 1.5× 1.5× 1.5 mm.

We select the kidney segmentation because of its articulated and relatively smooth shape. Figure 4.1 illustrates examples from the dataset. In order to remove corrupted or incomplete kidney segmentations, we apply the following filtering, which was empirically determined to remove obviously corrupted or incomplete segmentations. First, empty segmentation are discarded. Then, the number of 3D connected components is counted; if more than one component is found, the scan is discarded. Finally, segmentations containing voxels on the boundary of the volume are removed, as these indicate organs cropped at the scan boundaries, while those with fewer than 25,000 occupied voxels are discarded to exclude small or incomplete segmentations. This results in a total of 970 volumes of either the left or the right kidney, from which we randomly select 300 volumes. Kidney segmentations are centered and cropped to shape 85 × 85 × 110.

4.2.2 VerSe

The Large Scale Vertebrae Segmentation Challenge (VerSe) [SHB+21] is a collection of CT scans of the human spine, released as part of the MICCAI Vertebrae Segmentation Challenge. The dataset is collected from multiple medical centers and comprises 374 multi-detector CT scans from 355 patients, yielding 4505 manually verified individual vertebrae segmentations by radiology experts. Figure 4.2 depicts examples of individual vertebrae.

Following Amiranashvili et al. [ALL+22], this work filters all available CT scans and selects those with isotropically spaced voxels. All spine segmentations are then split into individual vertebrae, which results in a total dataset of 287 volumes. We centered and cropped each vertebra to shape 128 × 120 × 90.

Figure 4.1: Examples of kidney segmentations from the TotalSegmentator dataset [WBM+23].

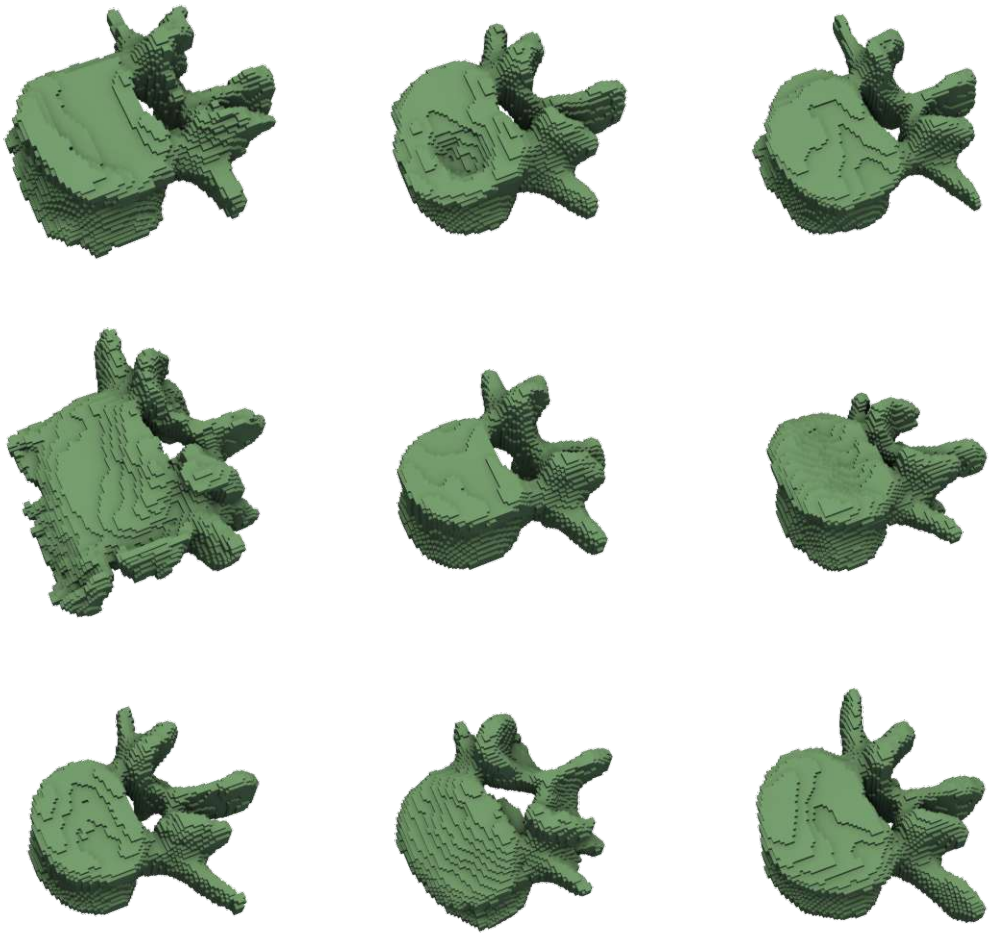


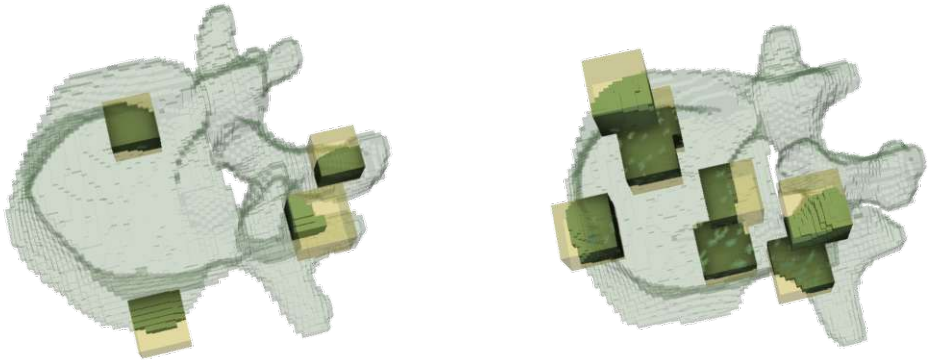
Figure 4.2: Examples of vertebrae segmentations from the VerSe dataset [SHB+21].

4.3 Baselines

Following the authors of influential sampling methods [DLA19, LMA20, YSA23], two non-learned sampling approaches are selected as a baseline: Random Sampling and Furthest Point Sampling.

Random Sampling

Random Sampling (RS) gathers samples at random. In our experiments, randomly selected samples are voxel-aligned within the input volume, and volumetric patches are extracted around them. Figure 4.3 illustrates an example of the RS applied to two vertebrae samples from the VerSe dataset [SHB+21].



(a) 4 patches of size $9 \times 9 \times 9$ (b) 8 patches of size $11 \times 11 \times 11$ voxels

Figure 4.3: Examples of the patches selected by the Random Sampling (RS) algorithm on the VerSe dataset [SHB+21].

Furthest Point Sampling

Furthest Point Sampling (FPS) is a widely adopted point cloud sampling method ensuring a well-distributed coverage of the input space [LMA20]. FPS iteratively selects a point that is the furthest from the set of previously selected points, and is fully deterministic, given the first point is selected in a fixed and consistent manner. To evaluate the generalization capabilities of the FPS, this work employs a non-deterministic variant of the FPS in which the first point is selected at random.

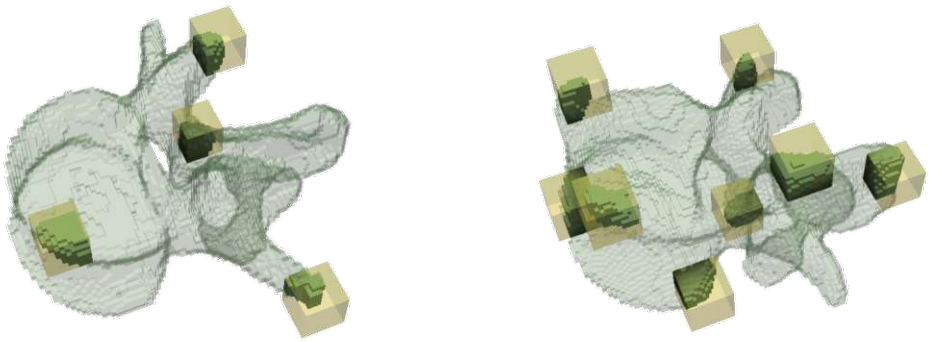
Similarly to RS, samples selected using FPS are voxel-aligned and patches around them are extracted. Figure 4.4 illustrates an example of the FPS applied to two vertebrae samples from the VerSe dataset [SHB+21].

4.4 Implementation Details

This section presents the implementation details of the proposed pipeline. Section 4.4.1 details the specific hyperparameter configuration, while Section 4.4.2 addresses the computational optimizations implemented to improve the efficiency of the reconstruction step.

4.4.1 Hyperparameters

We train our sampling pipeline using point clouds downsampled to $n = 2048$ points. This selection follows Dovrat et al. [DLA19] and Lang et al. [LMA20], who train their sampling architectures using a dataset with samples having exactly 2048 points. We



(a) 4 patches of size $9 \times 9 \times 9$ voxels (b) 8 patches of size $11 \times 11 \times 11$ voxels

Figure 4.4: Examples of the patches selected by the Furthest Point Sampling (FPS) algorithm on the VerSe dataset [SHB+21].

normalize every point cloud by mapping its spatial dimensions to the range $[-0.5, 0.5]$. Following Lang et al. [LMA20], our Samples Proposal Stage f_{gen}, depicted in Figure 4.5, consists of series of 1D Convolutional layers of sizes $[64, 128, 128, 256, 128]$, interleaved with ReLU activations and Batch Normalization, followed by a global max pool, and two Linear layers of size 256 with ReLU activation and Batch Normalizations. The Projection Stage f_{proj} represents each generated point as a weighted average of its $k = 32$ nearest neighbors, scaled by a learnable temperature parameter initialized as $t = 1.0$. Following Amiranashvili et al. [ALL+22], for each full-resolution volumetric reconstruction pass, a random latent vector is initialized as $z \sim \mathcal{N}(0, 10^{-4})$ and optimized using $T = 100$ optimization steps. The sampling architecture is trained by minimizing a weighted sum of the loss terms (Equations 3.13-3.18) with weights $\lambda_s = 1.0, \lambda_r = 1.0, \lambda_{proximity} = 1.0,$ and $\lambda_{repulsion} = 0.001$. We train the sampling architecture for 400 epochs using the Adam optimizer [KB14] with a batch size 8, and an initial learning rate of 0.01. A step learning rate scheduler is employed, reducing the learning rate by a factor of 10 after 200 epochs.

4.4.2 Gradient Tracking

The design of the reconstruction network proposed by Amiranashvili et al. [ALL+22] poses a challenge in implementing a memory-efficient backward pass that fits within a GPU memory. Each of the T iterations in the latent vector optimization produces an extensive computational graph. Additionally, the final pass over the full-resolution coordinate grid, described in Section 3.5.2, where gradients with respect to every input coordinate must be stored, requires substantial memory to retain intermediate results.

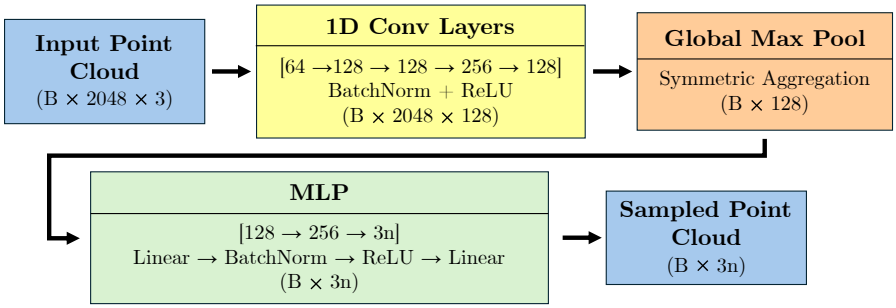


Figure 4.5: Overview of the utilized sampling architecture, which follows Lang et al. [LMA20].

To reduce the memory required for saving intermediate values, we employ two strategies. First, given a batch $B = \{K_i \mid i \in \{1, 2, ..., b\}\}$ of b patches, and a set of optimized latent vectors $Z = \{z^i_{rec} \mid z^i_{rec} \in \mathbb{R}^d, i \in \{1, 2, ...b\}\}$, each sample is processed individually in an explicit *for* loop. For each index i, the full resolution binary segmentation is reconstructed from K_i and its corresponding optimized latent vector z^i_{rec}. Afterwards, the reconstruction loss is computed, and gradients are backpropagated with respect to z^i_{rec}. Finally, the gradients obtained from all samples in the batch are accumulated and propagated through the preceding network layers.

Secondly, following Amiranashvili et al. [ALL+22], we split the full-resolution coordinate grid into batches, which further decreases the memory required and increases the throughput. With both strategies we are able to satisfy all memory requirements for a batched forward pass through the reconstruction network, store intermediate values and gradients, and backward them with respect to the patch coordinates.

Furthermore, to enable gradient tracking through the T steps of the latent vector optimization, we employ the library *higher* [GAY+19], which converts a neural network and its optimizer into differentiable, functional versions that support gradient tracking through the optimization steps.

4.5 Evaluation Setup

This section outlines the evaluation protocol used to assess the performance of all the compared methods. In Section 4.5.1, the metrics used to measure reconstruction accuracy and surface alignment are introduced. Section 4.5.2 describes the validation procedure in more detail for both learned and non-learned methods.

4.5.1 Evaluation Metrics

To evaluate reconstruction quality, this work adopts three metrics commonly used in the literature for 3D segmentation assessment [LJZ+18, KBDB+19, ALL+22, XWL+24]: the DICE score, the Average Surface Distance (ASD), and the Hausdorff Distance (HD).

DICE Score

The DICE Score is a measure used to evaluate the overlap between two binary volumes X and Y and is defined as [BSB+17]:

$$\mathrm{DSC}(X,Y) = \frac{2|X \cap Y|}{|X| + |Y|} \tag{4.1}$$

The DICE score ranges from 0 to 1, where a value of 1 indicates perfect agreement between the reconstructed and reference segmentations, and a value of 0 indicates no overlap.

Average Surface Distance

The ASD measures the average deviation of the surface of volume X from the surface of volume Y. In the context of volumetric binary segmentation, the surface of a segmentation is defined as the contour of the volume. ASD is formally defined as [BSB+17]:

$$\mathrm{ASD}(X,Y) = \frac{1}{|S_X| + |S_Y|} \left(\sum_{x \in S_X} d(x, S_Y) + \sum_{y \in S_Y} d(y, S_X) \right) \tag{4.2}$$

where:

- S_V is the set of surface voxels of volume V, defined as voxels with label 1 that have at least one neighboring voxel with label 0,
- $d(v, S)$ is the minimum Euclidean distance from voxel v to any voxel in the surface S, formally:
$$d(v, S) = \min_{s \in S} \|v - s\|_2$$

ASD is measured in physical units (e.g., millimeters). A lower ASD indicates closer surface alignment, with a value of 0 corresponding to a perfect surface match.

Hausdorff distance

The Hausdorff distance (HD) identifies the largest segmentation error between the surfaces of two segmentations, X and Y. The Hausdorff distance is defined as [BSB+17]:

$$\mathrm{HD}(X,Y) = \max\{\mathrm{hd}(S_X, S_Y), \mathrm{hd}(S_Y, S_X)\},$$

where:

- S_X and S_Y denote the sets of surface voxels of volumes X and Y, respectively,

- $\mathrm{hd}(A, B)$ is the *one-sided* Hausdorff distance from set A to set B, defined as:

$$\mathrm{hd}(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|_2.$$

Following the common practice in medical image segmentation [LJZ+18, KBDB+19, ALL+22, XWL+24], this work utilizes the robust Hausdorff distance, which uses a 95th percentile instead of the maximum to reduce sensitivity to outliers and small segmentation errors:

$$\mathrm{HD}_{95}(X, Y) = \max\{\mathrm{P}_{95}(\mathrm{hd}(S_X, S_Y)), \mathrm{P}_{95}(\mathrm{hd}(S_Y, S_X))\}, \tag{4.3}$$

where:

- $\mathrm{P}_{95}(\cdot)$ denotes the 95th percentile of the set of distances.

HD_{95} is measured in physical units (e.g., millimeters), with a value of 0 indicating a perfect prediction.

4.5.2 Validation

In order to provide a comprehensive and robust evaluation, avoid overfitting, and systematically evaluate algorithms based on randomness, validation plays a crucial role. Validation is the process of evaluating the performance of a model on unseen data to estimate its generalization ability and provide statistical robustness.

Both the FPS and the Random Sampling baseline approaches are non-deterministic: FPS selects the first point at random, while Random Sampling selects all the points at random. In order to gain better insight into the true performance of the baseline methods and ensure statistical robustness, this work validates each method by running it $n = 30$ times and averaging the evaluation metrics. This is a common practice in the evaluation of non-deterministic algorithms, which ensures that the reported performance is not dominated by random fluctuations [AB11].

To validate the learned sampling approaches, this work adopts a common practice in medical image segmentation by employing a 5-fold cross-validation scheme combined with a 70-10-20 train-validation-test split [RTH+22, TKMSH23, IWU+24, HDNK24]. For each dataset, all available data are split into 5 equally sized subsets (folds). Multiple models are trained, each on a distinct 4-fold (80% of the available data) and evaluated on the remaining fold (20%). Within the training subset, the data is further split into 70% for training and 10% for validation. Each model is therefore trained and evaluated on different subsets of the data, which supports hyperparameter tuning and avoids overfitting. The overall performance is reported as the mean metric computed across all test samples, obtained by aggregating the predictions of the models trained on the individual folds.

Dataset	# samples	train (70%)	val (10%)	test (20%)
VerSe [SHB+21]	287	200	28	59
TotalSegmentator [WBM+23]	300	210	30	60

Table 4.2: Number of samples in train-validation-test splits for benchmarked datasets.

Table 4.2 presents an overview of the number of samples in the training, validation, and test splits for both datasets used in this work.

To further assess the robustness of the proposed sampling approach, statistical hypothesis testing is employed, including the parametric t-test [Stu08] and the non-parametric Wilcoxon signed-rank test [Wil92].

CHAPTER 5

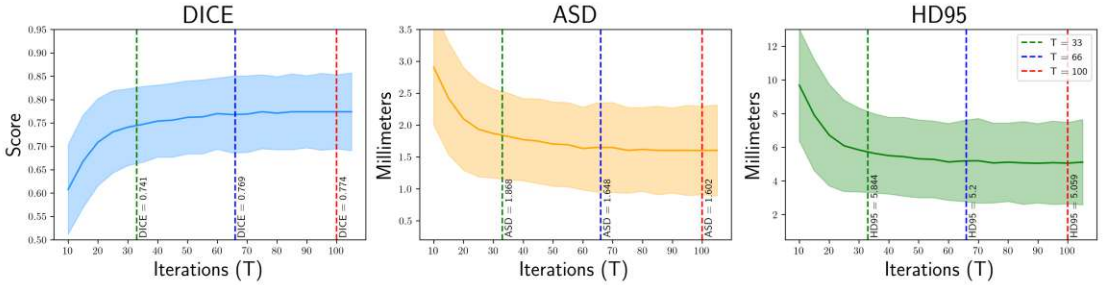
Results and Discussion

This chapter presents the evaluation results of the proposed sampling pipeline and provides a discussion and interpretation of the collected results. We evaluate the performance of our proposed learned sampling approach and compare it against the baseline FPS and Random Sampling methods described in Section 4.3. All methods are evaluated using the reconstruction metrics introduced in Section 4.5.1, following the experimental setup detailed in Section 4.5.

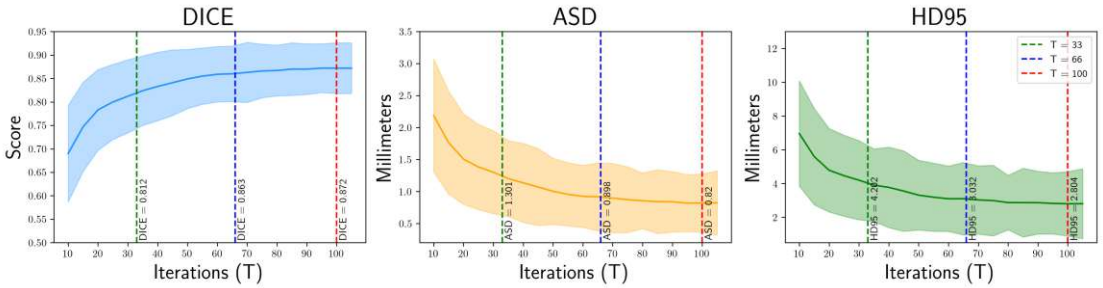
We organize the results according to the individual research questions and related experiments, and present them in separate sections. In Section 5.1, we analyze which patches are the most informative for the reconstruction task and how the quality of the reconstruction model affects the patches selected by our learned sampler, addressing RQ1. In Section 5.2, we tackle RQ2 by investigating how the patch configuration (i.e., the number of patches and the patch size) affects reconstruction performance metrics. Section 5.3 discusses the results of the experiments implemented to address RQ3 by evaluating computational aspects of the pipeline, including execution time and memory requirements of the proposed pipeline. In Section 5.4, we investigate the impact of selected hyperparameters used in the proposed sampling pipeline, and provide the results of an ablation study. Finally, in Section 5.5 we summarize the key findings and in Section 5.6 we discuss the limitations of the proposed sampling pipeline.

5.1 Patch Selection and Impact of the Reconstruction Model

In order to accurately reconstruct the full 3D shapes from partial observations, selected patches should capture the global nature of the object as well as internal structures. Patches placed on the surface of the shape are essential for correctly determining the size of the object, its orientation, and overall geometry. However, to reliably reconstruct complex topologies, such as those with internal cavities or holes, patches placed within the object are needed. To address RQ1, we evaluate which patches our learned sampler identifies as



(a) 4 patches of size $13 \times 13 \times 13$.



(b) 12 patches of size $13 \times 13 \times 13$.

Figure 5.1: Reconstruction performance evaluated for different numbers of latent vector optimization steps $T \in [10, 100]$. The green line represents $T = 33$, the blue $T = 66$, and the red $T = 100$, which are used in our further experiments.

most informative, such that reconstruction from these patches maximizes reconstruction performance. We compare the selected patches and resulting reconstructions with baseline sampling methods and analyze how changes in reconstruction network quality affect the patches selected by each sampling strategy. In our experiments, we evaluate reconstruction networks of low, medium, and high reconstruction quality, obtained by varying the number of optimization steps T (cf. Section 4.1). The effect of varying T on reconstruction performance is illustrated in Figure 5.1, where we analyze the reconstruction performance trends over the interval $T \in [10, 100]$ on two example patch configurations (4×13 and 12×13).

Across both datasets, the impact of the sampling strategy depends on reconstruction quality and object complexity: for low-quality reconstruction with $T = 33$ (cf. Tables 5.1, 5.4, and Figures 5.2a, 5.2b), the learned sampler provides increased DICE scores with improvements of approximately 2-3% in small to mid-size configurations ($N = \{4, 8\}$, $s = \{9, 11, 13\}$), along with consistent reductions in ASD (decrease by 0.2-0.5mm) and HD_{95} (decrease by 0.1-0.6mm) compared to FPS. This demonstrates that the proposed learned sampling is particularly effective when the reconstruction capacity is limited. For higher-quality reconstruction (cf. Tables 5.2, 5.3, 5.5, 5.6 and Figures 5.3, 5.4) with $T \in \{66, 100\}$, the effect of patch selection varies with object complexity. For complex vertebrae shapes,

the learned sampler continues to outperform baseline methods, with DICE improvements of 1.5–2% at $T = 66$ and 2-5% at $T = 100$ in small to mid-sized patch configurations. In both reconstruction settings, ASD decreases by 0.2-0.5mm and HD_{95} by 0.1-0.6mm. In the case of the Total Segmentator Kidneys dataset [WBM+23], which contains less complex kidney shapes, the importance of the sampling strategy decreases as the quality of the reconstruction network improves. At $T = 66$, DICE improvements compared to FPS are in the range 1-2%, with small to mid-size configurations. At $T = 100$, the evaluation metrics suggest comparable performance between the proposed sampling pipeline and the baseline FPS, with improvements in DICE within a 0-1% range.

We further illustrate the effect of reconstruction quality on patch selection in Figures 5.10 and 5.16, where the same input is sampled using different values of T. Figure 5.10 shows that the proposed sampler can adaptively place patches even inside the shape as needed, whereas FPS and RS cover primarily the surface of the shape. This behavior is visible in both figures, as repeated sampling of the same input produces different outputs (the value of T does not influence sampled patches).

Additionally, we analyze the spatial placement of the selected patches to better understand the differences observed across sampling strategies. We observe that capturing the full spatial extent of an object is essential for accurate reconstruction. This is illustrated in Figure 5.12, where the proposed learned sampler places two patches at the top of the kidney to cover its complete shape. In comparison, neither FPS nor RS selects a patch in this region, leading to missing voxels in the reconstruction for both baselines. However, sampling only on the surface of the shape is insufficient, especially for shapes with non-trivial topology, such as vertebrae. Here, a patch inside the object is needed to allow the reconstruction network to correctly reconstruct internal sections of the shape. As shown in Figure 5.10, neither FPS nor RS place patches inside the vertebra for any value of T, leading to a reconstruction that accurately captures the external shape but fails to reconstruct the internal hole. In comparison, the proposed learned sampler places patches inside the object and reconstructs the internal section of the shape accurately.

This highlights that the proposed learned sampler is capable of placing patches around the shape where needed. In contrast, FPS places patches based on maximal distance, i.e, maximal coverage, and is unable to place patches to the interior before the surface is sufficiently covered, limiting its ability to capture internal structures.

These experiments demonstrate that effective patch sampling is not simply a matter of uniform coverage, but a strategic choice between sampling the outer surface and the interior of a shape. The proposed sampling module is trained to place patches both along the boundaries and inside the structure, enabling more accurate reconstruction than baseline methods that lack this adaptivity.

Regarding RQ1, our findings show that the learned sampling component consistently identifies the most informative regions for volumetric reconstruction. Its effectiveness depends on both the quality of the reconstruction network and the complexity of the object. Improvements in evaluation metrics are particularly notable for vertebrae shapes

from VerSe [SHB+21], or when the reconstruction network is of lower quality. For TotalSegmentator Kindeys [WBM+23], the impact of sampling becomes less significant with increased quality of reconstruction. Nonetheless, the learned sampler achieves comparable DICE performance and additionally yields lower surface-based error than baselines.

5.2 Impact of Patch Configuration

Patch configuration, including the number and size of patches, directly influences the sampling strategy. A larger number of patches allows the sampler to distribute samples across the object, while larger patches provide broader contextual information. However, using a large number of large patches can be inefficient, computationally costly, and memory-intensive. For this reason, we analyze how patch configuration affects reconstruction quality and the resulting evaluation metrics, which addresses RQ2.

Tables 5.3 and 5.6 report DICE, ASD, and HD_{95} for VerSe [SHB+21] and Total Segmentator Kidneys [WBM+23] datasets, respectively. All experiments in this section were evaluated using the high-quality reconstruction network with $T = 100$ optimization steps, which we consider the default setting for the reconstruction evaluation (cf. Section 5.4.3). Figure 5.4 shows surface plots of the results in Tables 5.3 and 5.6 by presenting DICE scores as surface plots, allowing us to visualize trends in reconstruction performance across different patch configurations. Finally, we present visual results for both the proposed learned sampler and the baseline methods, including the sampled patches, the resulting reconstructions, and 2D and 3D comparisons highlighting reconstruction errors (Figures 5.5–5.15). For the qualitative analysis, we select five patch configurations, defined by the number of patches N and the patch size K ($N \times K$): 4×9, 4×13, 8×9, 12×11, and 12×13. These configurations were selected to cover a representative range of sampling scenarios, from a few small patches to many large ones, helping us assess the trade-off between the number of patches and their size.

We first analyze general trends in reconstruction performance across both datasets. Table 5.3 presents results for the VerSe [SHB+21] dataset. Overall, the learned sampler demonstrates the ability to select highly informative patches for the reconstruction task across all evaluation configurations. The benefits are particularly evident in small-to-mid-range configurations ($N = \{4, 8\}$, $s = \{9, 11, 13\}$), where the sampler consistently outperforms the baseline, with significant increases across all evaluation metrics. For example, with $N = 4$ patches of size $K = 9$, the proposed method achieves an improvement in DICE score of 5.7% over FPS and 5% over RS. Similarly, for the 8×9 configuration, the learned sampler outperforms FPS in DICE score by 3.3% and RS by 3.8%. This trend is clearly visible in the surface plot in Figure 5.4a, which shows a steep increase in DICE scores in the region of smaller to mid-size patch configurations. These results show that our proposed sampling method can select highly informative patches, leading to more accurate reconstructions with smaller inputs compared to baselines.

Table 5.6, reporting results for the Total Segmentator Kidneys dataset [WBM+23], shows

that for small to mid-size patch configurations, the proposed learned sampling algorithm achieves comparable reconstruction performance to FPS, and outperforms Random Sampling consistently across all evaluated configurations. The results from Table 5.6 are further supported by the surface plot in Figure 5.4b, which highlights a similar trend in reconstruction performance between the proposed approach and FPS.

Next, we investigate the performance of individual patch configurations for the proposed learned sampling method, as reported in Tables 5.3 and 5.6. In many cases, it is preferable to select a larger number of smaller patches rather than fewer larger ones. For the VerSe dataset [SHB+21], this is illustrated by, e.g., the 8×9 configuration, which achieves better results than 4×13 (DICE 0.813 vs. 0.800) despite being approximately 35% smaller in the number of voxels, and by the 12×11 configuration, which outperforms 8×13 (DICE 0.836 vs. 0.835) while having roughly 9% fewer voxels. Additionally, the 12×9 configuration yields a higher DICE Score than 4×13 (0.828 vs. 0.800), despite both configurations having approximately the same number of voxels. This trend continues in the Total Segmentator Kidneys dataset [WBM+23], where similarly the 8×9 configuration outperforms 4×13 (DICE 0.866 vs. 0.863).

From the visual analysis, we observe that the learned sampler adapts its sampling strategy to the number of patches available. For small patch counts, the sampler tends to distribute samples around the shape to capture its overall structure (e.g., proposed patches in Figures 5.5 and 5.6). As the number of patches increases, samples start to appear within the interior of the shape (e.g., proposed patches in Figure 5.7). For high patch counts, the sampler densely covers all interior regions (e.g., proposed patches in Figures 5.8 and 5.9). In comparison, baseline sampling does not adapt its strategy to the number of available patches or the underlying shape, and instead provides inconsistent results due to randomness. To illustrate this behavior, we refer to Figure 5.10. The figure shows the patches selected by each sampling strategy for different values of T, where each strategy was provided with identical input. Baseline samplers, for which the patch selection does not depend on T, exhibit non-deterministic behavior, leading to substantially different patch selections across runs. This variability is reflected in the reconstruction quality: for instance, Random Sampling (RS) with a low-quality reconstruction significantly outperforms its medium-quality counterpart (DICE 0.711 vs. 0.558), even though both operate on the same input data.

In conclusion, our results show that reconstruction quality strongly depends on patch size and number of patches (RQ2). The proposed learning sampling component is especially effective for small-to-mid-size patch configurations, where improvements over the selected baseline range from 1% to 5%. A further analysis of individual patch configurations reveals that, in many cases, it is advantageous to select a larger number of smaller patches, as such configurations achieve comparable or better reconstruction performance than configurations using fewer larger patches.

VerSe (low-quality reconstruction)										
		DICE↑			**ASD↓**			**HD_{95} ↓**		
N	K	Proposed	FPS	RS	Proposed	FPS	RS	Proposed	FPS	RS
4	9	**0.741 ± 0.07**	0.712 ± 0.06	0.697 ± 0.06	**1.952 ± 0.66**	2.199 ± 0.53	2.317 ± 0.60	**6.343 ± 2.54**	6.937 ± 1.99	7.519 ± 2.38
4	11	**0.741 ± 0.07**	0.714 ± 0.06	0.689 ± 0.06	**1.895 ± 0.68**	2.093 ± 0.51	2.313 ± 0.65	**6.070 ± 2.68**	6.568 ± 1.98	7.591 ± 2.62
4	13	**0.763 ± 0.07**	0.744 ± 0.06	0.729 ± 0.07	**1.674 ± 0.60**	1.843 ± 0.55	1.969 ± 0.63	**5.309 ± 2.36**	5.723 ± 1.96	6.302 ± 2.38
8	9	**0.773 ± 0.07**	0.749 ± 0.06	0.742 ± 0.06	**1.657 ± 0.60**	1.864 ± 0.50	1.903 ± 0.56	**5.450 ± 2.42**	5.798 ± 1.86	6.031 ± 2.19
8	11	**0.778 ± 0.07**	0.758 ± 0.06	0.753 ± 0.06	**1.587 ± 0.63**	1.757 ± 0.52	1.791 ± 0.56	**5.234 ± 2.49**	5.375 ± 1.91	5.593 ± 2.16
8	13	**0.797 ± 0.06**	0.773 ± 0.06	0.770 ± 0.07	**1.454 ± 0.60**	1.661 ± 0.58	1.675 ± 0.60	**4.788 ± 2.34**	5.068 ± 2.10	5.226 ± 2.21
12	9	**0.795 ± 0.07**	0.766 ± 0.06	0.760 ± 0.06	**1.466 ± 0.61**	1.715 ± 0.55	1.750 ± 0.56	**4.888 ± 2.52**	5.294 ± 1.90	5.397 ± 1.99
12	11	**0.795 ± 0.06**	0.786 ± 0.06	0.772 ± 0.06	**1.442 ± 0.57**	1.523 ± 0.54	1.614 ± 0.55	**4.731 ± 2.10**	4.910 ± 1.88	5.159 ± 1.93
12	13	0.817 ± 0.07	**0.819 ± 0.07**	0.806 ± 0.07	1.253 ± 0.62	**1.246 ± 0.60**	1.336 ± 0.62	4.168 ± 2.34	**4.063 ± 2.18**	4.355 ± 2.28

Table 5.1: Comparison of evaluation metrics between the proposed method and the FPS and RS baselines on the VerSe dataset [SHB+21] using the low-quality ($T = 33$) reconstruction network. N denotes the number of selected patches, and K the shape of each patch. **Bold** indicates the best result per metric. **Blue bold** indicates a statistically significant difference between the proposed method and FPS, with the better-performing method highlighted. Statistical significance was assessed using both the paired t-test and the Wilcoxon signed-rank test with $p < 0.05$ ($\alpha = 0.05$).

VerSe (medium-quality reconstruction)										
		DICE↑			**ASD↓**			**HD_{95} ↓**		
N	K	Proposed	FPS	RS	Proposed	FPS	RS	Proposed	FPS	RS
4	9	**0.749 ± 0.07**	0.705 ± 0.06	0.707 ± 0.06	**1.917 ± 0.66**	2.284 ± 0.54	2.251 ± 0.56	**6.308 ± 2.67**	7.242 ± 1.94	7.319 ± 2.11
4	11	**0.762 ± 0.07**	0.726 ± 0.05	0.716 ± 0.05	**1.733 ± 0.61**	2.037 ± 0.50	2.100 ± 0.56	**5.640 ± 2.31**	6.280 ± 1.67	6.815 ± 2.03
4	13	**0.788 ± 0.06**	0.769 ± 0.06	0.756 ± 0.06	**1.481 ± 0.54**	1.640 ± 0.54	1.751 ± 0.57	**4.750 ± 2.07**	5.154 ± 1.88	5.708 ± 2.09
8	9	**0.797 ± 0.06**	0.774 ± 0.06	0.769 ± 0.06	**1.468 ± 0.54**	1.679 ± 0.54	1.699 ± 0.51	**4.995 ± 2.24**	5.318 ± 1.80	5.436 ± 1.86
8	11	**0.802 ± 0.06**	0.794 ± 0.06	0.781 ± 0.06	**1.395 ± 0.52**	1.467 ± 0.54	1.565 ± 0.55	4.683 ± 2.07	**4.633 ± 1.98**	5.011 ± 2.03
8	13	**0.822 ± 0.07**	0.821 ± 0.05	0.806 ± 0.06	**1.255 ± 0.59**	1.267 ± 0.52	1.381 ± 0.54	4.336 ± 2.44	**4.031 ± 1.95**	4.527 ± 2.05
12	9	0.812 ± 0.06	**0.811 ± 0.06**	0.798 ± 0.06	**1.329 ± 0.58**	1.349 ± 0.53	1.446 ± 0.52	4.453 ± 2.15	**4.308 ± 1.82**	4.648 ± 1.89
12	11	0.826 ± 0.06	**0.839 ± 0.05**	0.816 ± 0.06	1.195 ± 0.52	**1.087 ± 0.49**	1.266 ± 0.51	4.078 ± 1.86	**3.589 ± 1.86**	4.233 ± 1.93
12	13	0.851 ± 0.06	**0.862 ± 0.05**	0.843 ± 0.06	0.987 ± 0.60	**0.904 ± 0.49**	1.050 ± 0.53	3.491 ± 2.30	**3.055 ± 1.92**	3.609 ± 2.09

Table 5.2: Comparison of evaluation metrics between the proposed method and the FPS and RS baselines on the VerSe dataset [SHB+21] using the medium-quality ($T = 66$) reconstruction network. N denotes the number of selected patches, and K the shape of each patch. **Bold** indicates the best result per metric. **Blue bold** indicates a statistically significant difference between the proposed method and FPS, with the better-performing method highlighted. Statistical significance was assessed using both the paired t-test and the Wilcoxon signed-rank test with $p < 0.05$ ($\alpha = 0.05$).

VerSe (high-quality reconstruction)										
		DICE↑			ASD↓			HD_{95} ↓		
N	K	Proposed	FPS	RS	Proposed	FPS	RS	Proposed	FPS	RS
4	9	**0.758 ± 0.06**	0.701 ± 0.06	0.708 ± 0.06	**1.835 ± 0.60**	2.337 ± 0.56	2.252 ± 0.57	**6.130 ± 2.35**	7.461 ± 2.01	7.386 ± 2.12
4	11	**0.766 ± 0.06**	0.730 ± 0.06	0.724 ± 0.06	**1.697 ± 0.59**	2.003 ± 0.55	2.039 ± 0.54	**5.577 ± 2.17**	6.207 ± 1.77	6.689 ± 1.97
4	13	**0.800 ± 0.07**	0.775 ± 0.06	0.765 ± 0.06	**1.385 ± 0.59**	1.589 ± 0.53	1.684 ± 0.55	**4.545 ± 2.20**	5.046 ± 1.87	5.553 ± 2.03
8	9	**0.813 ± 0.05**	0.780 ± 0.06	0.775 ± 0.06	**1.334 ± 0.45**	1.629 ± 0.54	1.648 ± 0.51	**4.577 ± 1.84**	5.231 ± 1.81	5.348 ± 1.86
8	11	**0.817 ± 0.06**	0.802 ± 0.06	0.791 ± 0.06	**1.279 ± 0.54**	1.395 ± 0.52	1.483 ± 0.51	**4.484 ± 2.28**	4.442 ± 1.91	4.817 ± 1.93
8	13	**0.835 ± 0.05**	0.830 ± 0.05	0.817 ± 0.06	**1.137 ± 0.43**	1.191 ± 0.47	1.294 ± 0.52	**4.004 ± 1.92**	3.856 ± 1.79	4.352 ± 2.03
12	9	**0.828 ± 0.05**	0.820 ± 0.06	0.807 ± 0.06	**1.204 ± 0.94**	1.274 ± 0.52	1.371 ± 0.51	4.231 ± 2.24	**4.101 ± 1.82**	4.485 ± 1.83
12	11	0.836 ± 0.06	**0.850 ± 0.05**	0.829 ± 0.05	1.110 ± 0.54	**0.993 ± 0.45**	1.159 ± 0.47	3.820 ± 2.10	**3.309 ± 1.76**	3.941 ± 1.81
12	13	0.862 ± 0.06	**0.872 ± 0.05**	0.854 ± 0.06	0.914 ± 0.54	**0.822 ± 0.45**	0.962 ± 0.50	3.282 ± 2.06	**2.820 ± 1.83**	3.381 ± 1.98

Table 5.3: Comparison of evaluation metrics between the proposed method and the FPS and RS baselines on the VerSe dataset [SHB+21] using the high-quality ($T = 100$) reconstruction network. N denotes the number of selected patches, and K the shape of each patch. **Bold** indicates the best result per metric. **Blue bold** indicates a statistically significant difference between the proposed method and FPS, with the better-performing method highlighted. Statistical significance was assessed using both the paired t-test and the Wilcoxon signed-rank test with $p < 0.05$ ($\alpha = 0.05$).

Total Segmentator Kidneys (low-quality reconstruction)										
		DICE↑			**ASD↓**			**HD_{95} ↓**		
N	K	Proposed	FPS	RS	Proposed	FPS	RS	Proposed	FPS	RS
4	9	**0.811 ± 0.05**	0.795 ± 0.05	0.780 ± 0.05	**1.573 ± 0.58**	1.771 ± 0.51	1.824 ± 0.58	**4.742 ± 1.76**	5.078 ± 1.56	5.602 ± 1.76
4	11	**0.823 ± 0.05**	0.804 ± 0.05	0.793 ± 0.05	**1.469 ± 0.61**	1.691 ± 0.56	1.727 ± 0.58	**4.508 ± 1.92**	4.953 ± 1.79	5.354 ± 1.81
4	13	**0.828 ± 0.06**	0.810 ± 0.05	0.801 ± 0.05	**1.438 ± 0.63**	1.633 ± 0.59	1.662 ± 0.61	**4.296 ± 1.84**	4.739 ± 1.85	5.137 ± 1.87
8	9	**0.827 ± 0.06**	0.792 ± 0.05	0.802 ± 0.05	**1.450 ± 0.60**	1.736 ± 0.56	1.620 ± 0.59	**4.607 ± 1.90**	5.136 ± 1.85	4.978 ± 1.83
8	11	**0.839 ± 0.05**	0.823 ± 0.04	0.822 ± 0.05	**1.334 ± 0.57**	1.478 ± 0.52	1.457 ± 0.55	**4.085 ± 1.74**	4.332 ± 1.60	4.484 ± 1.70
8	13	**0.856 ± 0.05**	0.842 ± 0.04	0.839 ± 0.05	**1.192 ± 0.56**	1.341 ± 0.54	1.339 ± 0.55	**3.765 ± 1.81**	4.133 ± 1.76	4.242 ± 1.75
12	9	**0.849 ± 0.05**	0.842 ± 0.05	0.839 ± 0.05	**1.278 ± 0.62**	1.369 ± 0.57	1.358 ± 0.57	**4.062 ± 1.99**	4.194 ± 1.86	4.258 ± 1.79
12	11	**0.862 ± 0.05**	0.853 ± 0.05	0.849 ± 0.05	**1.130 ± 0.60**	1.222 ± 0.55	1.236 ± 0.55	**3.548 ± 1.88**	3.773 ± 1.83	3.917 ± 1.78
12	13	**0.880 ± 0.04**	0.874 ± 0.04	0.865 ± 0.04	**0.964 ± 0.49**	1.026 ± 0.51	1.097 ± 0.52	**3.128 ± 1.50**	3.304 ± 1.59	3.548 ± 1.58

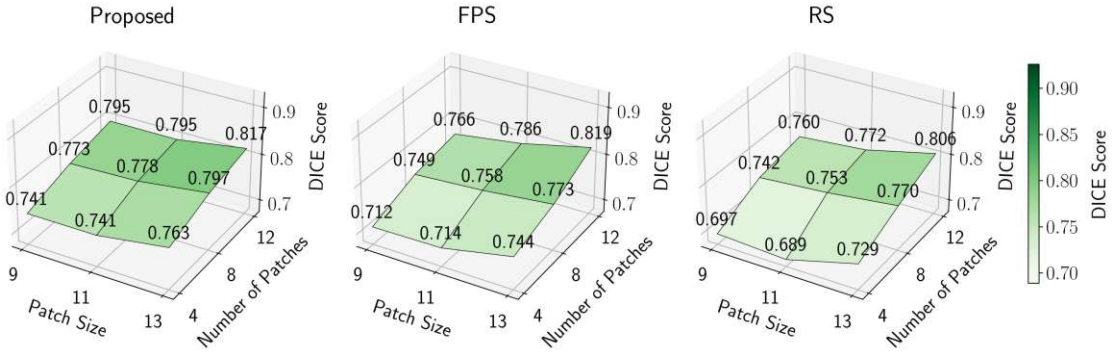
Table 5.4: Comparison of evaluation metrics between the proposed method and the FPS and RS baselines on the Total Segmentator Kidneys dataset [WBM+23] using the low-quality ($T = 33$) reconstruction network. N denotes the number of selected patches, and K the shape of each patch. **Bold** indicates the best result per metric. **Blue bold** indicates a statistically significant difference between the proposed method and FPS, with the better-performing method highlighted. Statistical significance was assessed using both the paired t-test and the Wilcoxon signed-rank test with $p < 0.05$ ($\alpha = 0.05$).

Total Segmentator Kidneys (medium-quality reconstruction)										
		DICE↑			**ASD↓**			**HD_{95} ↓**		
N	K	Proposed	FPS	RS	Proposed	FPS	RS	Proposed	FPS	RS
4	9	0.823 ± 0.05	**0.831 ± 0.04**	0.803 ± 0.04	1.509 ± 0.58	**1.437 ± 0.46**	1.638 ± 0.51	4.765 ± 1.95	**4.530 ± 1.49**	5.288 ± 1.60
4	11	**0.846 ± 0.05**	0.839 ± 0.05	0.817 ± 0.04	**1.280 ± 0.58**	1.357 ± 0.54	1.520 ± 0.52	4.257 ± 1.96	4.454 ± 1.84	5.016 ± 1.70
4	13	**0.852 ± 0.05**	0.851 ± 0.04	0.831 ± 0.04	**1.233 ± 0.58**	1.243 ± 0.48	1.403 ± 0.52	**4.045 ± 1.94**	4.064 ± 1.61	4.661 ± 1.66
8	9	**0.857 ± 0.05**	0.854 ± 0.04	0.838 ± 0.04	**1.164 ± 0.50**	1.182 ± 0.50	1.323 ± 0.50	**3.991 ± 1.76**	4.141 ± 1.88	4.433 ± 1.74
8	11	**0.877 ± 0.04**	**0.877 ± 0.03**	0.859 ± 0.03	0.989 ± 0.40	**0.984 ± 0.37**	1.138 ± 0.40	**3.350 ± 1.38**	3.435 ± 1.35	3.855 ± 1.30
8	13	**0.890 ± 0.03**	**0.890 ± 0.03**	0.873 ± 0.03	**0.886 ± 0.45**	0.894 ± 0.40	1.033 ± 0.42	**3.086 ± 1.67**	3.314 ± 1.56	3.658 ± 1.50
12	9	0.885 ± 0.03	**0.889 ± 0.03**	0.871 ± 0.03	0.942 ± 0.44	**0.924 ± 0.41**	1.067 ± 0.43	**3.274 ± 1.58**	3.542 ± 1.68	3.752 ± 1.50
12	11	0.898 ± 0.03	**0.900 ± 0.03**	0.884 ± 0.03	0.796 ± 0.40	**0.776 ± 0.38**	0.920 ± 0.41	**2.781 ± 1.40**	2.891 ± 1.46	3.255 ± 1.42
12	13	0.911 ± 0.02	**0.915 ± 0.03**	0.898 ± 0.03	0.677 ± 0.29	**0.645 ± 0.31**	0.790 ± 0.35	**2.411 ± 1.08**	2.491 ± 1.29	2.869 ± 1.18

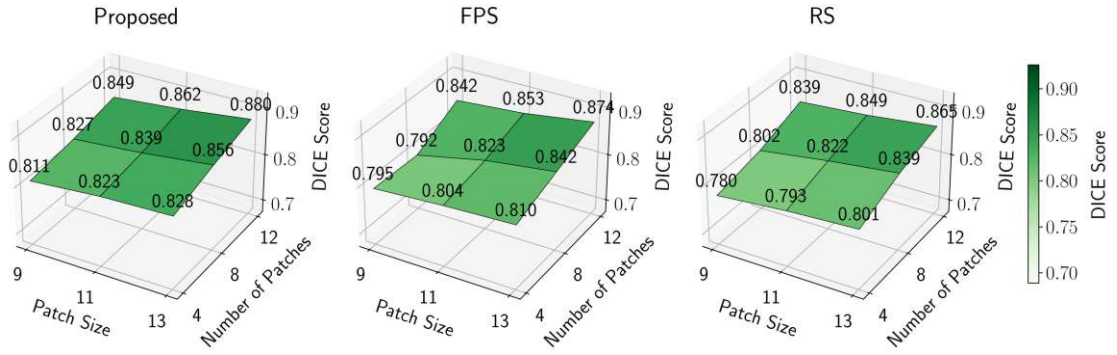
Table 5.5: Comparison of evaluation metrics between the proposed method and the FPS and RS baselines on the Total Segmentator Kidneys dataset [$WBM^{+}23$] using the medium-quality ($T = 66$) reconstruction network. N denotes the number of selected patches, and K the shape of each patch. **Bold** indicates the best result per metric. **Blue bold** indicates a statistically significant difference between the proposed method and FPS, with the better-performing method highlighted. Statistical significance was assessed using both the paired t-test and the Wilcoxon signed-rank test with $p < 0.05$ ($\alpha = 0.05$).

Total Segmentator Kidneys (high-quality reconstruction)										
		DICE↑			**ASD↓**			**HD_{95} ↓**		
N	K	Proposed	FPS	RS	Proposed	FPS	RS	Proposed	FPS	RS
4	9	**0.842 ± 0.04**	0.840 ± 0.04	0.812 ± 0.04	**1.311 ± 0.41**	1.346 ± 0.42	1.560 ± 0.46	**4.364 ± 1.47**	4.380 ± 1.41	5.148 ± 1.48
4	11	**0.849 ± 0.04**	**0.849 ± 0.04**	0.825 ± 0.04	**1.256 ± 0.47**	1.267 ± 0.48	1.453 ± 0.48	4.346 ± 1.70	**4.311 ± 1.70**	4.904 ± 1.57
4	13	**0.863 ± 0.04**	0.862 ± 0.04	0.840 ± 0.04	**1.124 ± 0.44**	1.144 ± 0.42	1.319 ± 0.48	**3.897 ± 1.64**	3.914 ± 1.48	4.522 ± 1.62
8	9	0.866 ± 0.04	**0.869 ± 0.04**	0.850 ± 0.04	1.102 ± 0.50	**1.054 ± 0.44**	1.219 ± 0.44	3.901 ± 1.90	**3.866 ± 1.72**	4.197 ± 1.54
8	11	**0.891 ± 0.03**	0.890 ± 0.03	0.870 ± 0.03	**0.860 ± 0.33**	0.869 ± 0.30	1.042 ± 0.34	**3.130 ± 1.36**	3.235 ± 1.21	3.665 ± 1.17
8	13	0.901 ± 0.03	**0.902 ± 0.03**	0.884 ± 0.03	0.786 ± 0.36	**0.778 ± 0.31**	0.927 ± 0.34	**2.951 ± 1.52**	3.089 ± 1.30	3.407 ± 1.28
12	9	0.892 ± 0.03	**0.900 ± 0.03**	0.881 ± 0.03	0.875 ± 0.34	**0.819 ± 0.31**	0.964 ± 0.34	**3.303 ± 1.56**	3.396 ± 1.50	3.528 ± 1.28
12	11	0.910 ± 0.02	**0.913 ± 0.03**	0.896 ± 0.03	0.688 ± 0.30	**0.663 ± 0.29**	0.813 ± 0.33	**2.518 ± 1.17**	2.620 ± 1.20	3.002 ± 1.26
12	13	0.924 ± 0.02	**0.926 ± 0.02**	0.910 ± 0.02	0.553 ± 0.23	**0.545 ± 0.23**	0.685 ± 0.26	**2.049 ± 0.99**	2.255 ± 1.11	2.604 ± 0.96

Table 5.6: Comparison of evaluation metrics between the proposed method and the FPS and RS baselines on the Total Segmentator Kidneys dataset [WBM+23] using the high-quality ($T = 100$) reconstruction network. N denotes the number of selected patches, and K the shape of each patch. **Bold** indicates the best result per metric. **Blue bold** indicates a statistically significant difference between the proposed method and FPS, with the better-performing method highlighted. Statistical significance was assessed using both the paired t-test and the Wilcoxon signed-rank test with $p < 0.05$ ($\alpha = 0.05$).



(a) VerSe [SHB+21]



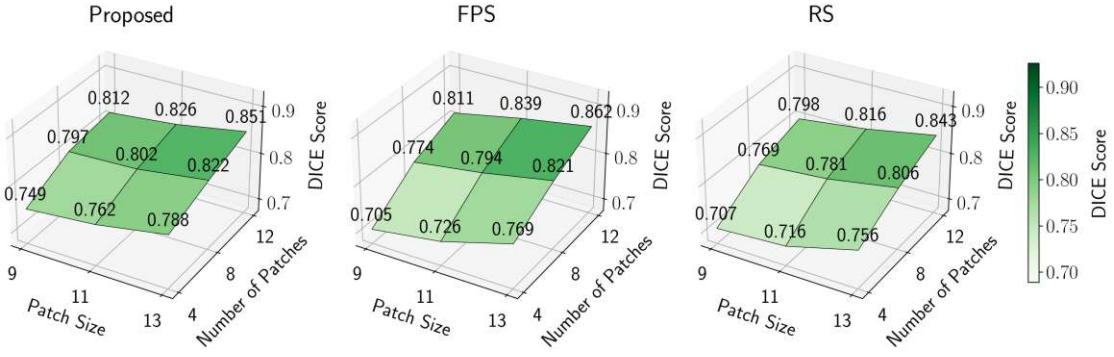
(b) Total Segmentator Kidneys [WBM+23]

Figure 5.2: DICE Scores for Total Segmentator Kidneys [WBM+23] and VerSe [SHB+21] across different patch configurations using low-quality ($T = 33$) reconstruction network.

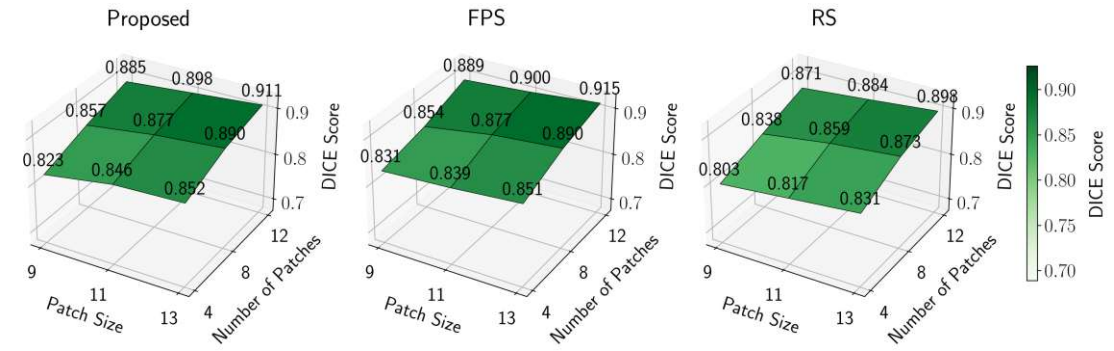
5.3 Computational and Memory Demands

The proposed sampling pipeline has the potential to accelerate the processing of large medical volumetric scans by significantly reducing the amount of data that must be analyzed, sampling only a subset of the input voxel grid, and reconstructing the full volumetric scan from the selected patches. We address RQ3 by measuring execution times and GPU memory usage for different patch configurations. Section 5.3.1 reports the execution times and memory requirements of the learned sampling module and includes a comparison against baseline sampling strategies. Section 5.3.2 presents the corresponding execution times and GPU memory usage of the reconstruction network. Finally, the practical implications of these results are discussed in Section 5.3.3, illustrating how the proposed sampling pipeline can be applied in environments with limited computational resources.

In each section, we report the computational demands of the inference process, and do not include the memory consumed by the model parameters themselves. Memory and



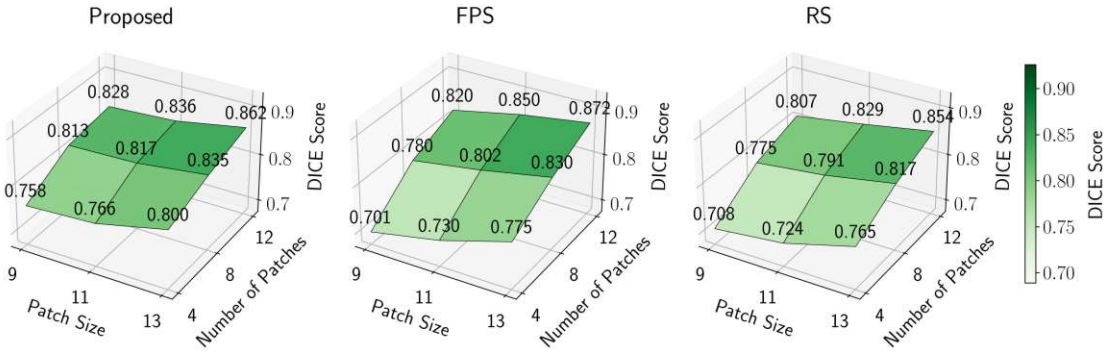
(a) VerSe [SHB+21]



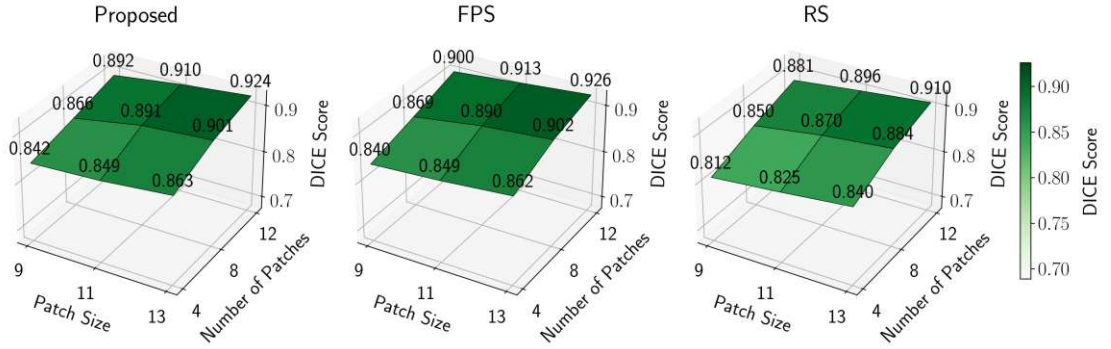
(b) Total Segmentator Kidneys [WBM+23]

Figure 5.3: DICE Scores for Total Segmentator Kidneys [WBM+23] and VerSe [SHB+21] across different patch configurations using medium-quality ($T = 66$) reconstruction network.

parameter counts for each model are provided separately in Table 5.7. Additionally, Table 5.7 provides the number of parameters and memory usage for three different patch configurations, showing that variations in patch number and size have only a little effect on the overall memory footprint of the sampling architecture.



(a) VerSe [SHB+21]



(b) Total Segmentator Kidneys [WBM+23]

Figure 5.4: DICE Scores for Total Segmentator Kidneys [WBM+23] and VerSe [SHB+21] across different patch configurations using high-quality ($T = 100$) reconstruction network.

Component	Number of Parameters	Memory [MB]
Sampling (4 patches of size $9 \times 9 \times 9$)	212,109	0.81
Sampling (8 patches of size $11 \times 11 \times 11$)	215,193	0.82
Sampling (12 patches of size $13 \times 13 \times 13$)	218,277	0.83
Reconstruction	116,609	0.44

Table 5.7: Number of parameters and memory for the sampling module and the reconstruction network. The variation in patch configuration for the sampling module results in only minimal differences in parameters and memory required.

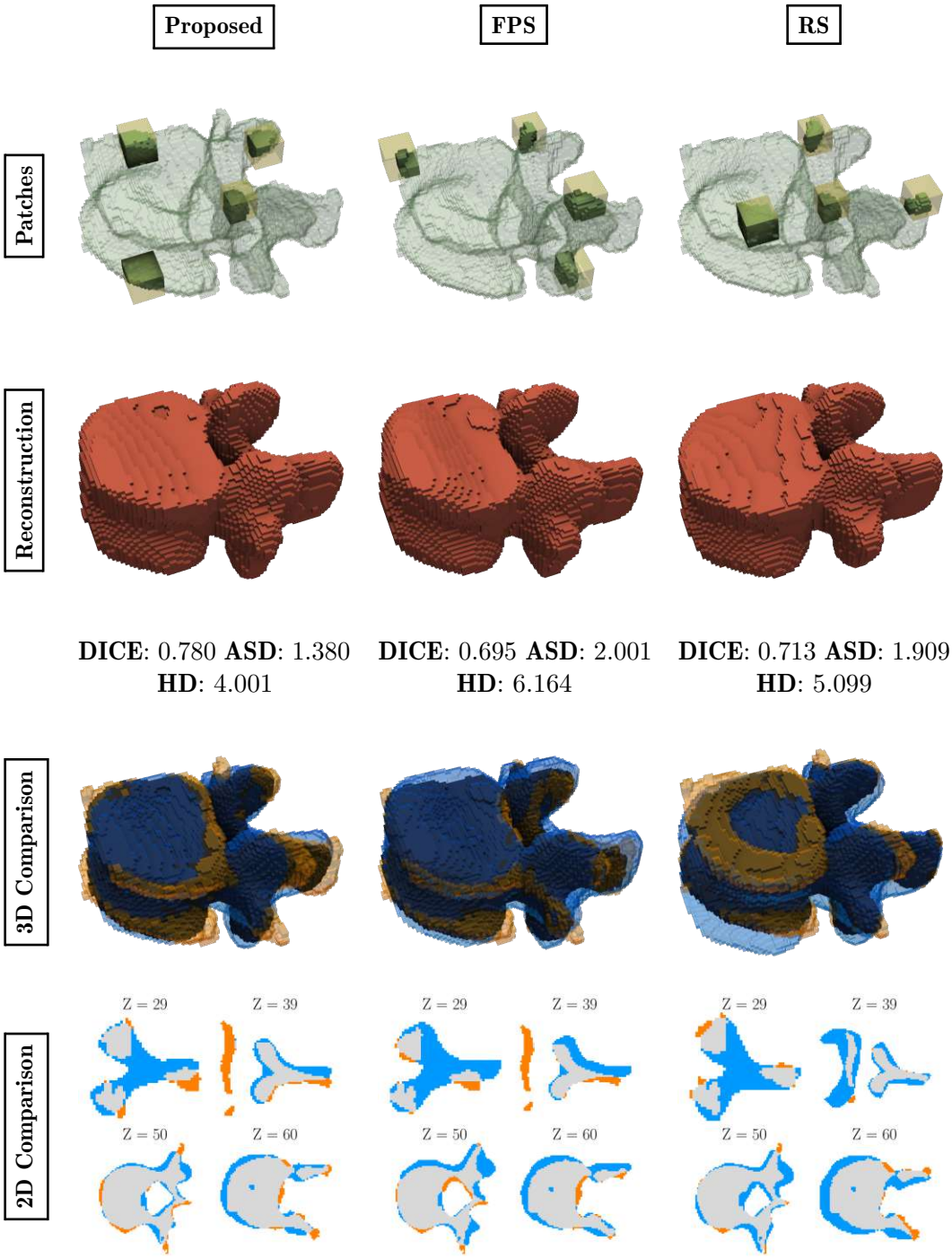


Figure 5.5: Qualitative results on VerSe [SHB+21] using $N = 4$, $K = 9$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 90$).

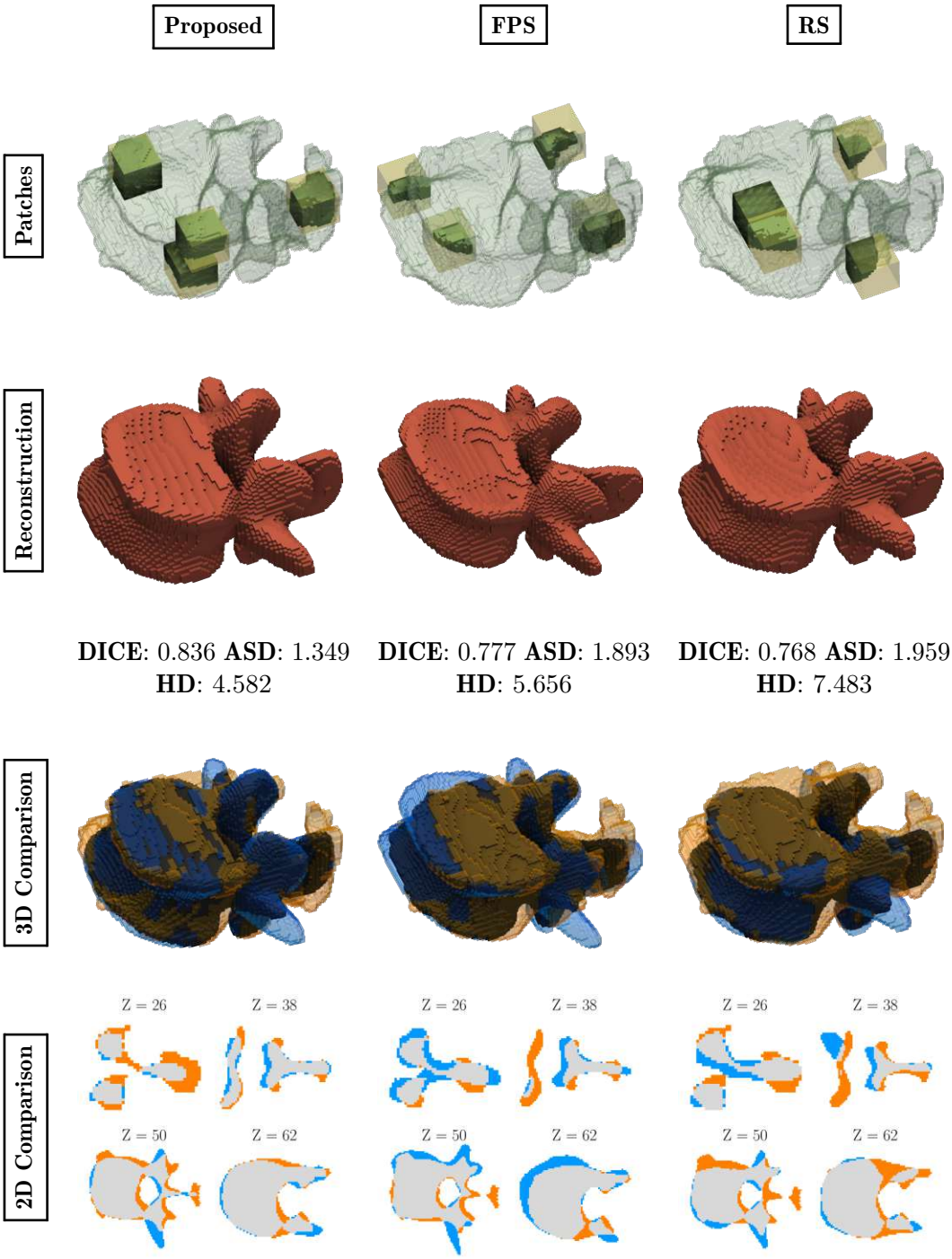


Figure 5.6: Qualitative results on VerSe [SHB+21] using $N = 4$, $K = 13$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 90$).

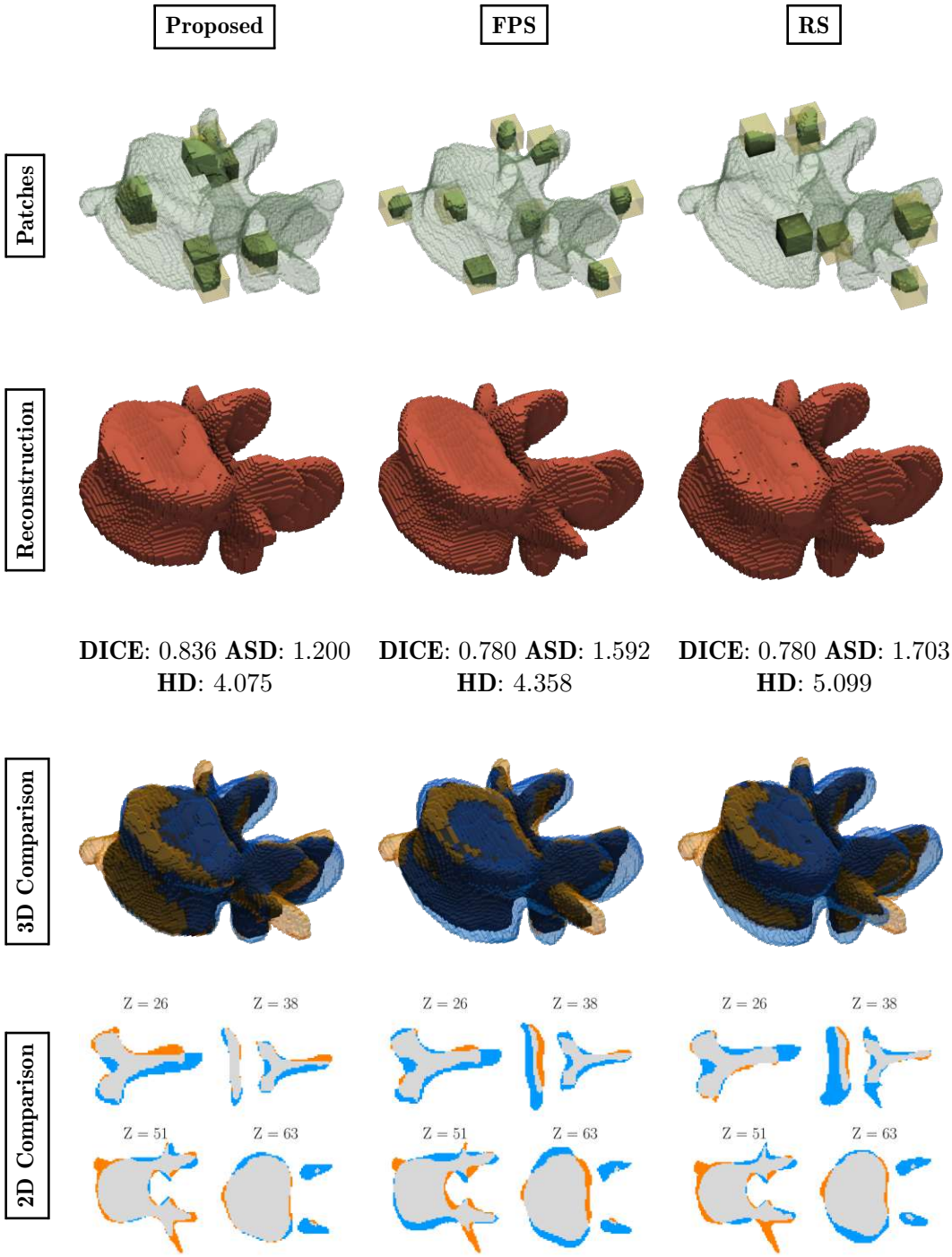


Figure 5.7: Qualitative results on VerSe [SHB+21] using $N = 8$, $K = 9$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 90$).

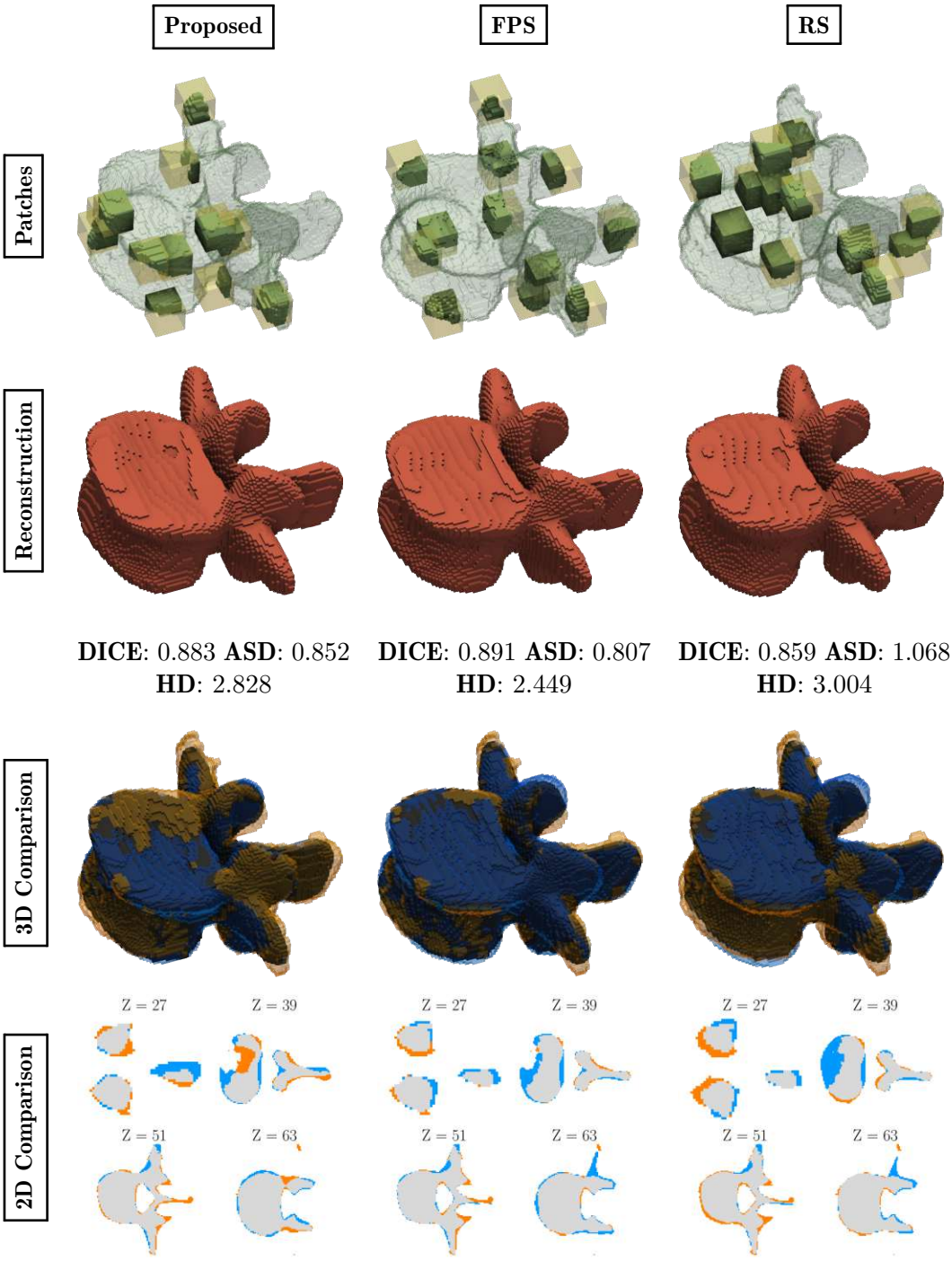


Figure 5.8: Qualitative results on VerSe [SHB+21] using $N = 12$, $K = 11$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 90$).

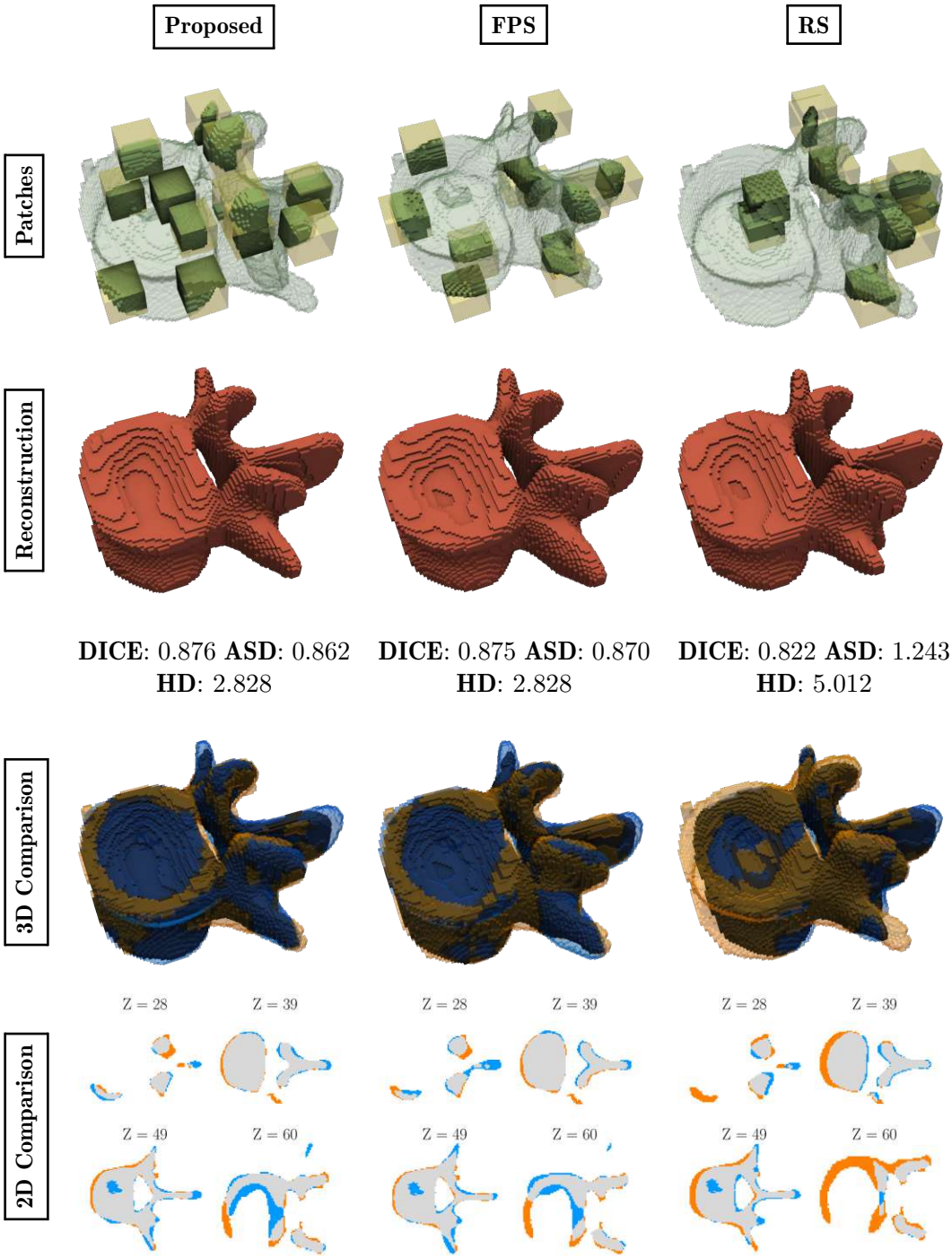


Figure 5.9: Qualitative results on VerSe [SHB+21] using $N = 12$, $K = 13$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 90$).

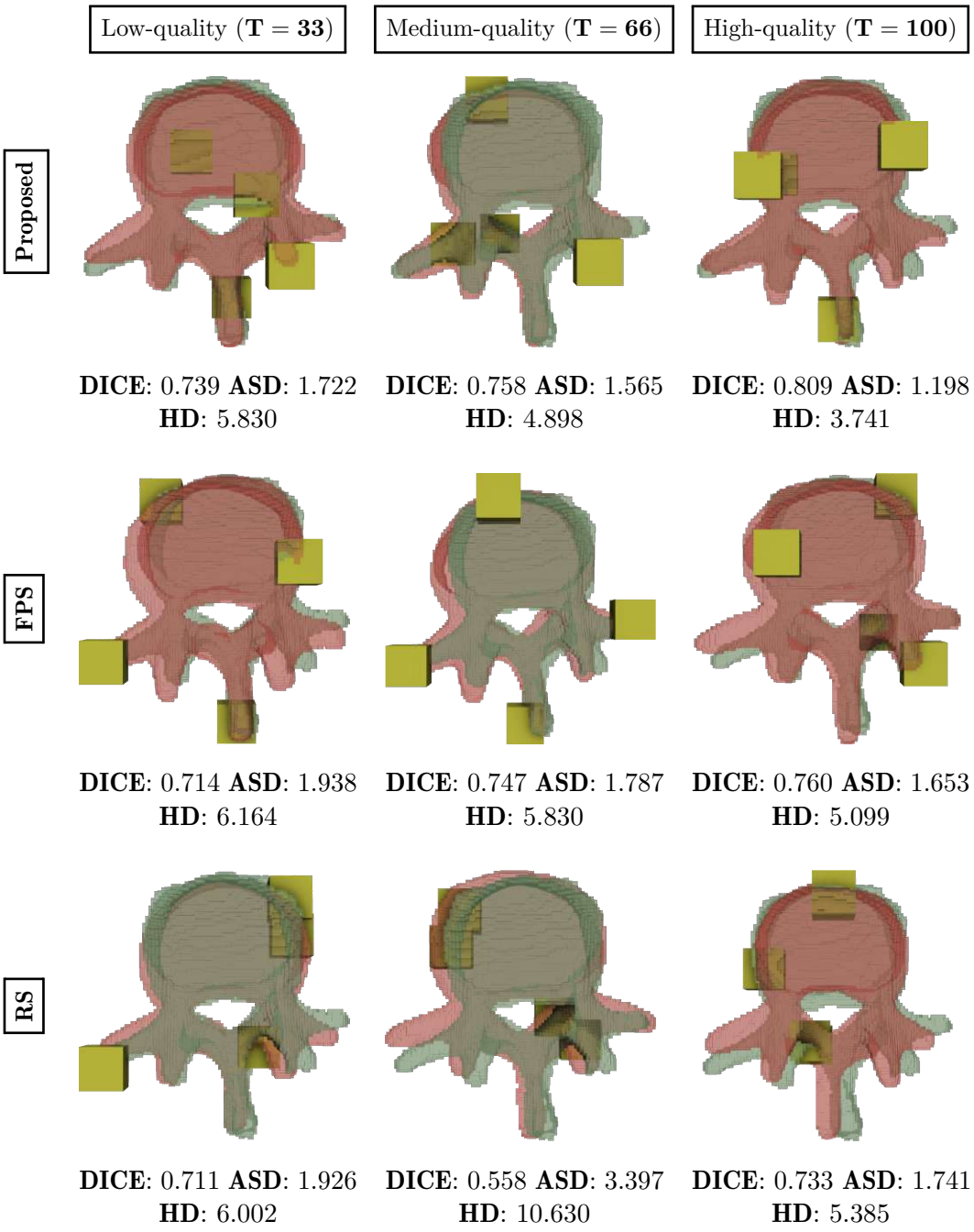


Figure 5.10: Comparison of patches sampled by proposed learned sampler and baseline methods under different reconstruction settings on the VerSe dataset [SHB+21]. Green: ground truth, Red: reconstruction, Yellow: selected volumetric patches. Patch configuration $N = 4$, $K = 13$.

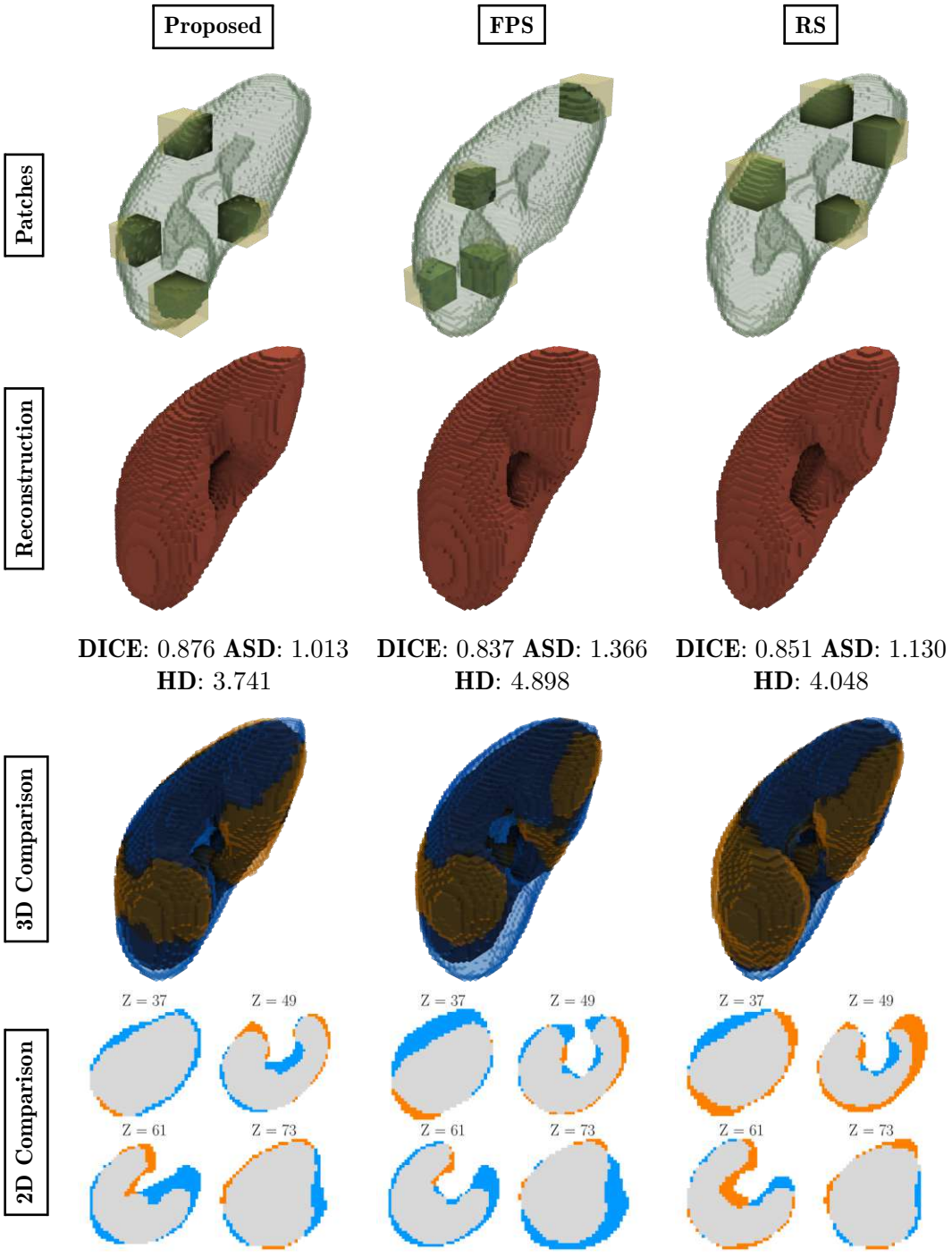


Figure 5.11: Qualitative results on Total Segmentator Kidneys [WBM+23] using $N = 4$, $K = 9$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 110$).

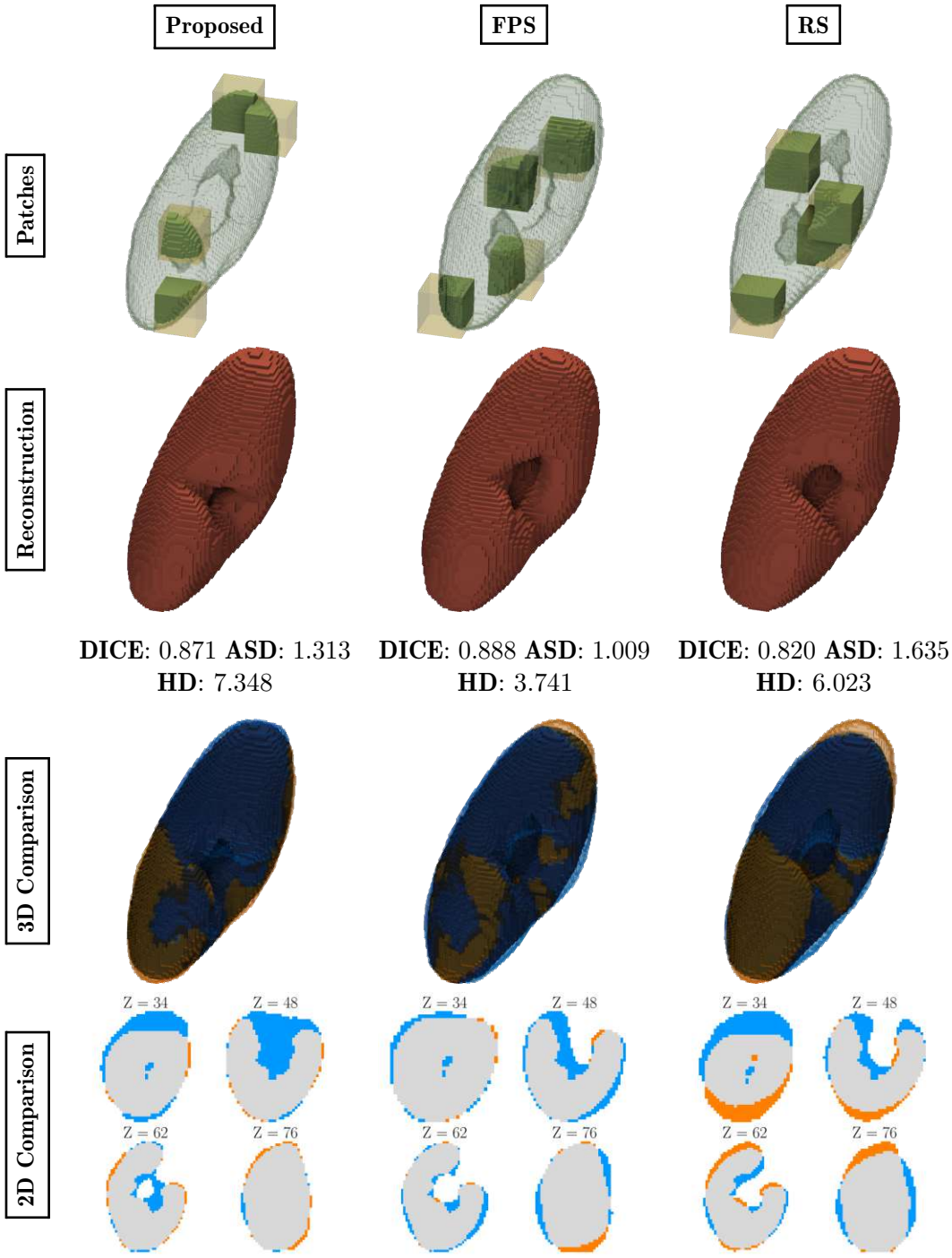


Figure 5.12: Qualitative results on Total Segmentator Kidneys [WBM+23] using $N = 4$, $K = 13$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 110$).

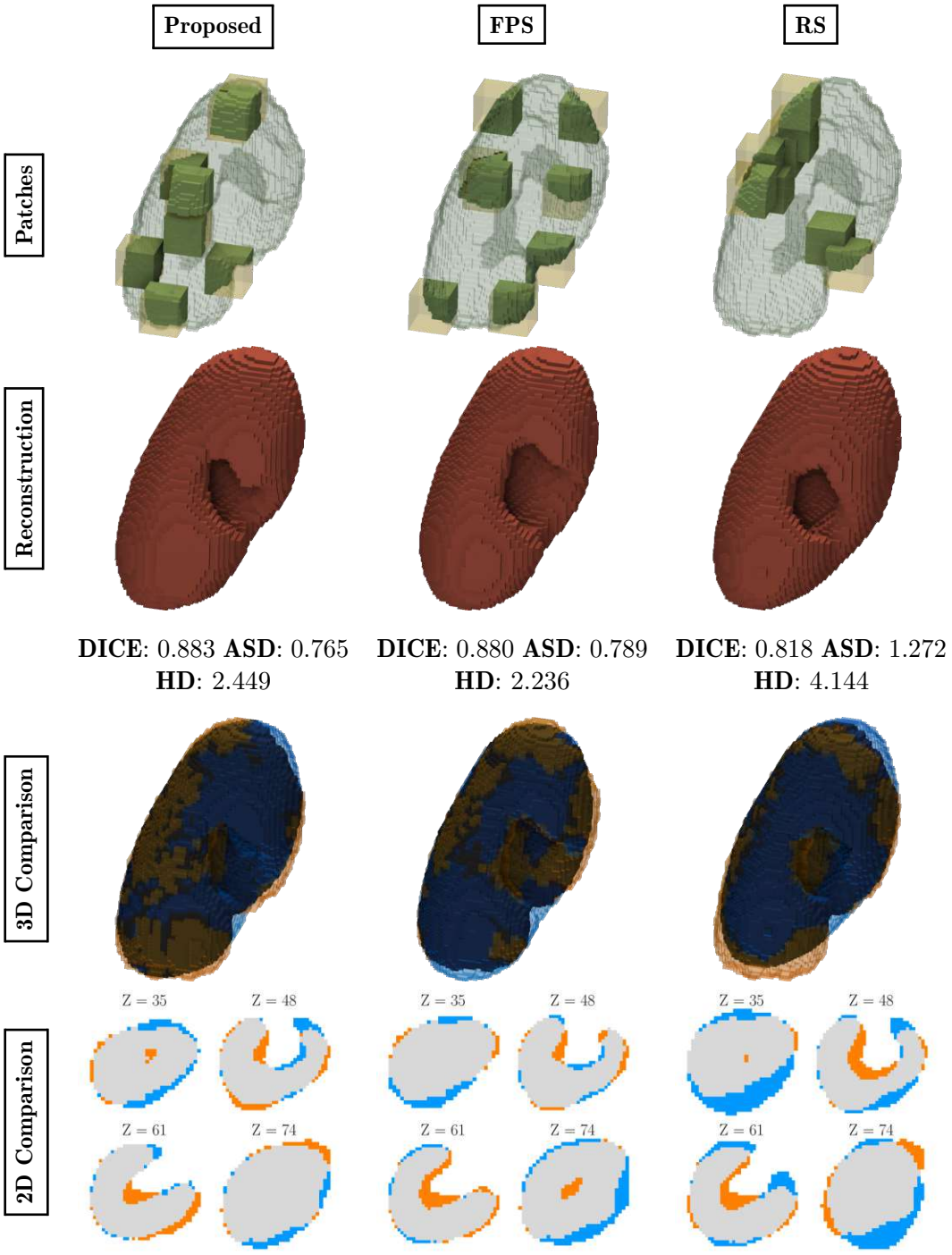


Figure 5.13: Qualitative results on Total Segmentator Kidneys [WBM+23] using $N = 8$, $K = 9$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 110$).

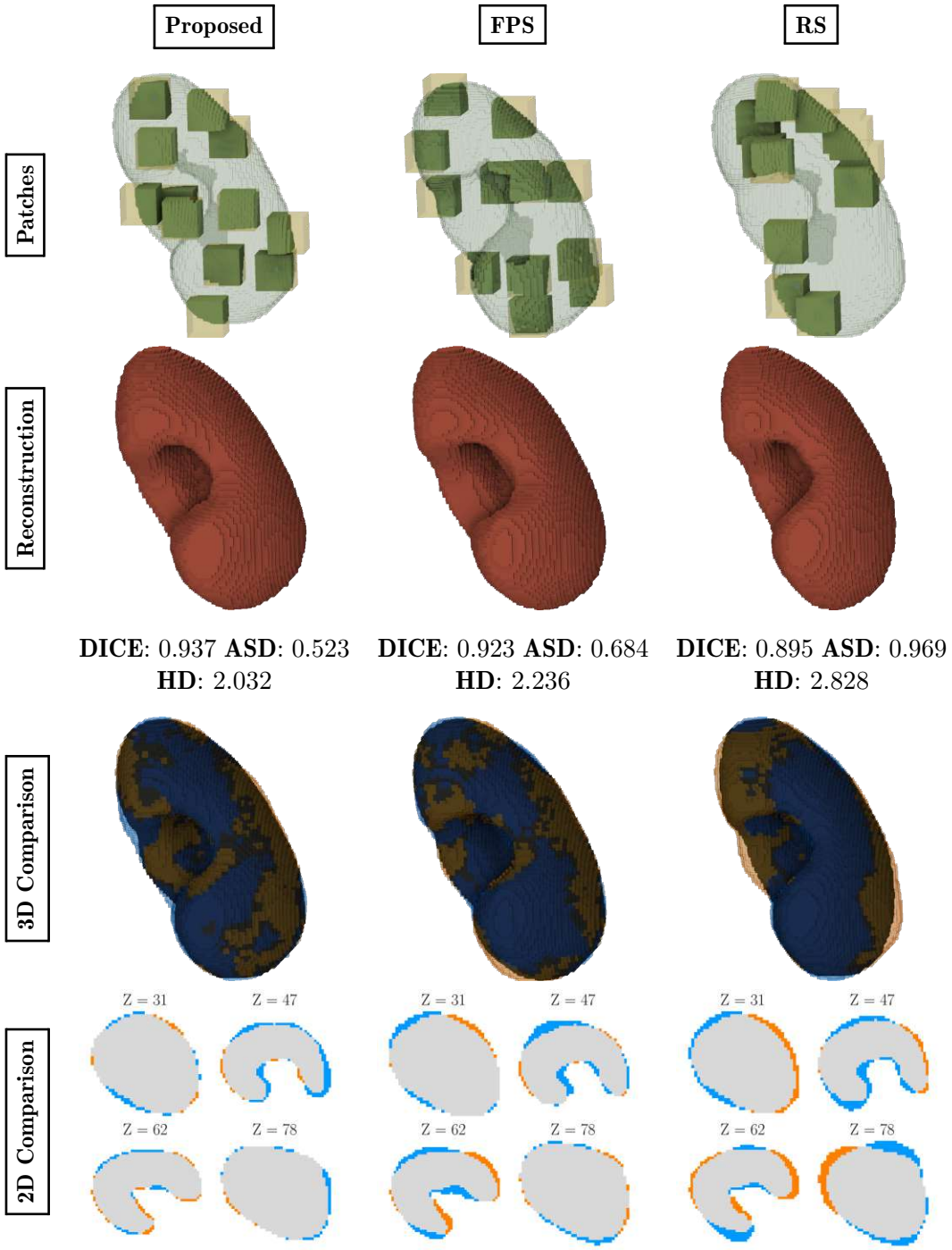


Figure 5.14: Qualitative results on Total Segmentator Kidneys [WBM+23] using $N = 12$, $K = 11$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 110$).

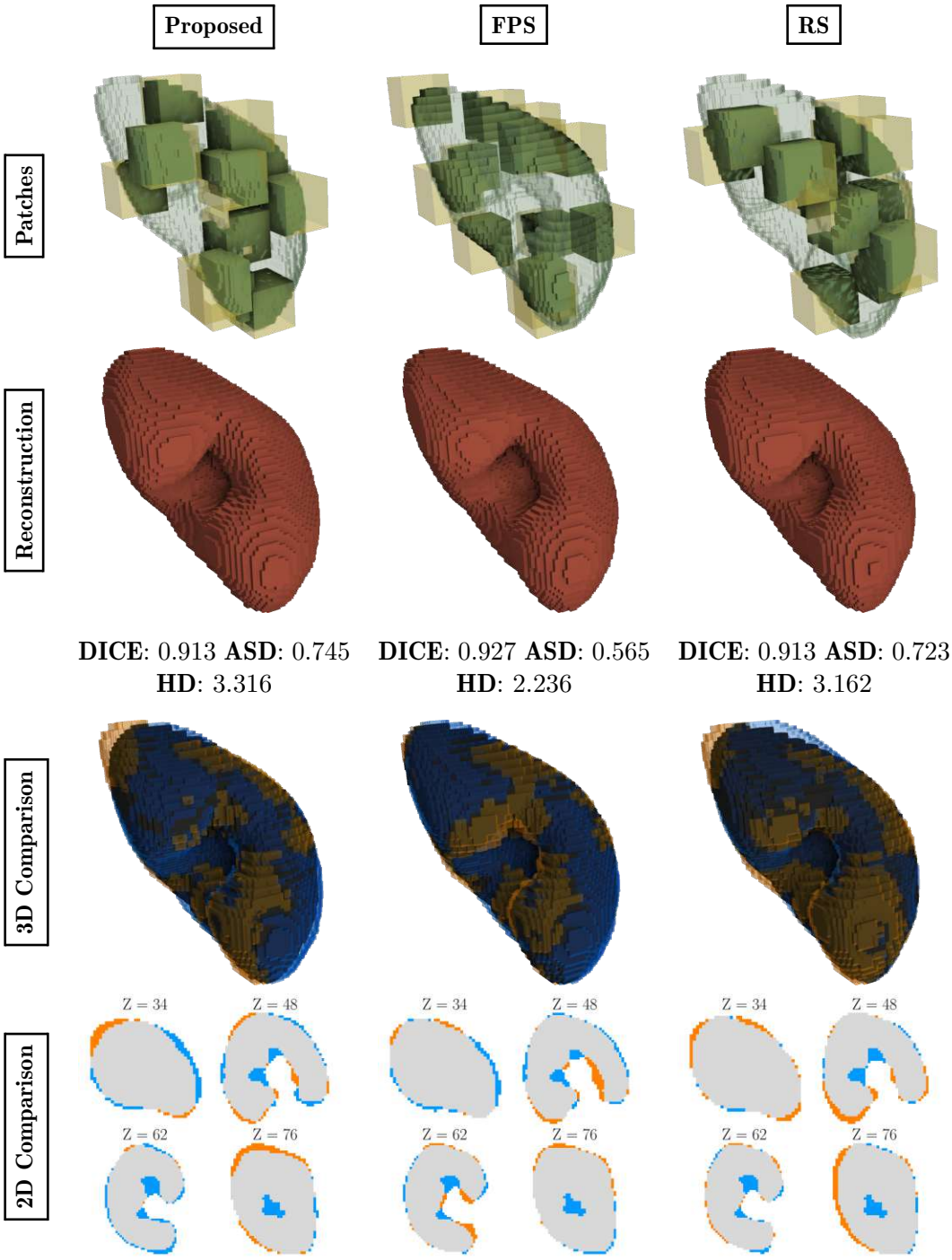


Figure 5.15: Qualitative results on Total Segmentator Kidneys [WBM+23] using $N = 12$, $K = 13$, and $T = 100$. From left to right: proposed learned sampler, FPS, and RS. Blue indicates extra reconstructed voxels, orange denotes missing voxels, and gray indicates the intersection. Z refers to the slice index along the z axis (top: $Z = 0$, bottom: $Z = 110$).

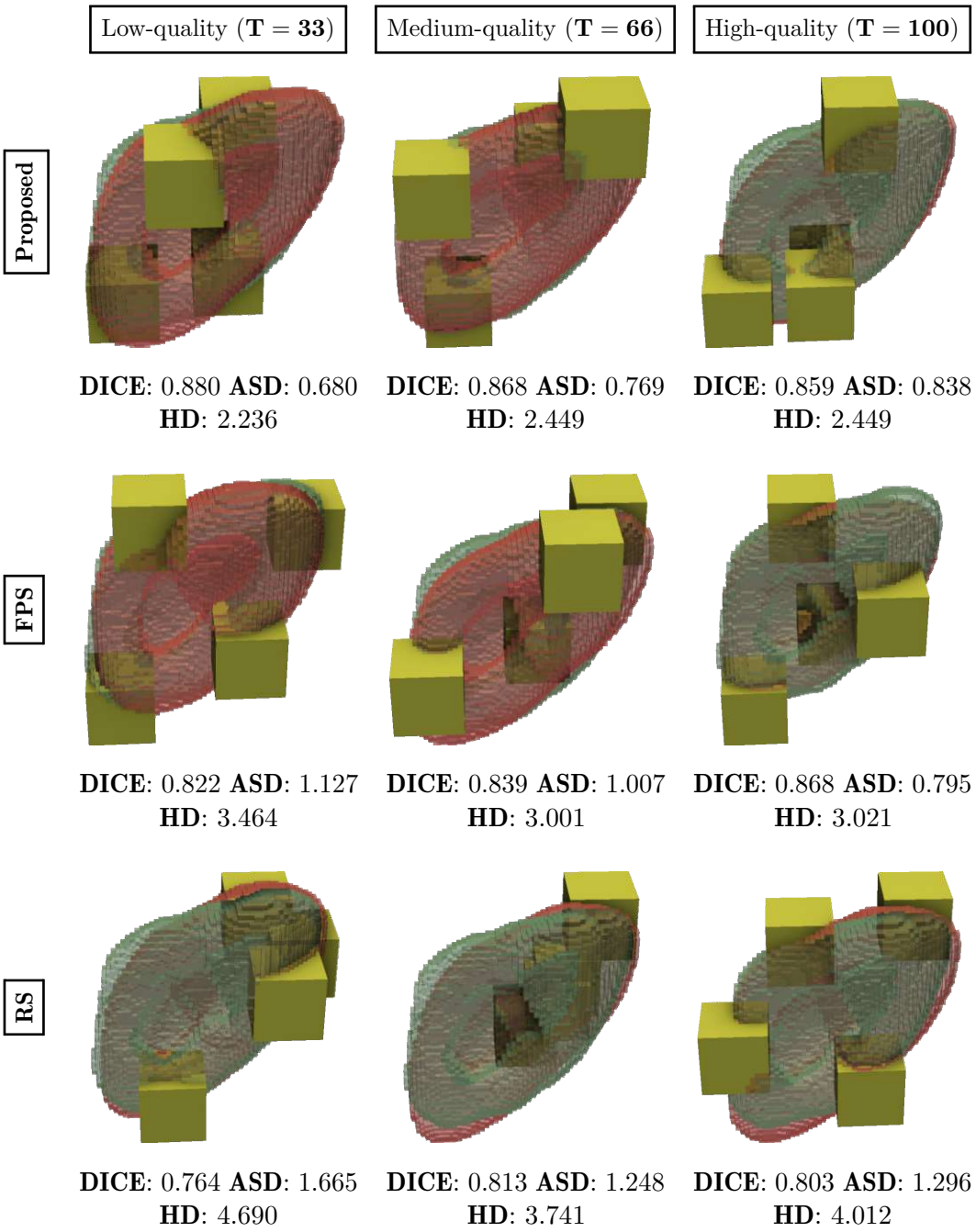


Figure 5.16: Comparison of patches sampled by proposed learned sampler and baseline methods under different reconstruction settings on the Total Segmentator Kidneys dataset [WBM+23]. Green: ground truth, Red: reconstruction, Yellow: selected volumetric patches. Patch configuration $N = 4$, $K = 13$.

Patch Config		Inference Time [ms]			Peak Memory Usage [MB]		
N	K	Our	FPS	RS	Our	FPS	RS
4	9	1.766 ± 0.50	1.056 ± 0.65	0.613 ± 0.64	168.0	44.1	44.1
4	11	1.943 ± 0.14	0.906 ± 0.07	0.529 ± 0.04	168.1	44.2	44.2
4	13	1.879 ± 0.25	0.828 ± 0.07	0.528 ± 0.03	168.1	44.4	44.4
8	9	1.886 ± 0.16	0.822 ± 0.07	0.591 ± 0.05	168.1	44.3	44.3
8	11	2.003 ± 0.21	0.921 ± 0.08	0.528 ± 0.03	168.2	44.6	44.6
8	13	1.840 ± 0.21	0.860 ± 0.08	0.529 ± 0.04	168.3	45.0	45.0
12	9	1.938 ± 0.15	0.957 ± 0.22	0.547 ± 0.06	168.1	44.4	44.4
12	11	2.116 ± 0.36	0.913 ± 0.06	0.529 ± 0.04	168.2	44.9	44.9
12	13	2.042 ± 0.22	0.917 ± 0.07	0.531 ± 0.03	168.4	45.5	45.5

Table 5.8: Inference times and memory requirements of the proposed sampling module and baseline methods

5.3.1 Sampling

We analyze the computational requirements of our sampling method from the point at which the downsampled point cloud is obtained until the volumetric patches are extracted. Operations that are independent of the sampling method, such as the Voxel Grid Conversion with downsampling, are not included, as their cost is constant. The computational cost of the sampling procedure depends only on the selected patch configuration, as all datasets are converted to point clouds of identical size. For this reason, Table 5.8 reports inference time and peak GPU memory usage with respect to the selected patch configuration only.

Our results indicate that differences in patch configuration have minimal impact on execution time and peak memory usage, due to the lightweight design of the sampling architecture. This efficiency makes it feasible to run the sampler even on machines with very limited resources.

5.3.2 Reconstruction

We evaluate the computational requirements of the reconstruction model, including both the optimization of the shape-specific latent vector z and the subsequent full-resolution inference using the optimized vector z_{rec}. Execution time and memory usage for the reconstruction depend only on the patch configuration and target resolution, and do not vary with the selected sampling method. For this reason, Table 5.9 reports the execution times and memory usage, shown for different patch configurations and the target resolutions of $85 \times 85 \times 110$ for the TotalSegmentator Kidneys dataset [WBM+23] and $128 \times 120 \times 90$ for the VerSe dataset [SHB+21].

Patch Config		Total Segmentator [WBM+23] (shape 85 × 85 × 110)		VerSe [SHB+21] (shape 128 × 120 × 90)	
N	K	Time [ms]	Memory [MB]	Time [ms]	Memory [MB]
4	9	392.6 ± 14.5	425.8	405.1 ± 12.3	436.3
4	11	389.7 ± 13.4	425.9	404.9 ± 8.7	436.4
4	13	386.7 ± 9.6	426.0	408.5 ± 9.4	436.5
8	9	373.0 ± 13.8	425.9	389.2 ± 8.7	436.4
8	11	367.0 ± 8.5	426.1	389.2 ± 9.3	436.6
8	13	366.8 ± 8.8	426.4	388.2 ± 9.4	436.9
12	9	385.4 ± 11.7	426.0	405.1 ± 9.5	436.6
12	11	383.4 ± 9.2	426.3	405.4 ± 8.9	436.9
12	13	386.0 ± 8.7	426.8	403.2 ± 8.7	437.3

Table 5.9: Execution times and memory usage of the reconstruction network for the benchmarking datasets. N refers to the number of selected patches, while K symbols the size of each patch

5.3.3 Deployment of our Sampling Pipeline

Results from Section 5.3.1 and 5.3.2 indicate that it is possible to deploy the proposed sampling pipeline in environments with limited computational resources, as the combined memory footprint of the sampling architecture and the reconstruction network remains moderate for both our benchmarking datasets. This improved efficiency is achieved by processing only the selected patches rather than the full volumetric scan, which substantially reduces the memory required by the processing network. For comparison, Chen et al. [CML+24] report that current end-to-end segmentation networks require between 3.81 and 13.91 GB of memory to process two patches of size $96 \times 96 \times 96$. In contrast, the smallest patch configuration we evaluated (4 patches of size $9 \times 9 \times 9$) contains approximately 3,000 voxels, corresponding to 0.33% of a $96 \times 96 \times 96$ patch, while the largest configuration (12 patches of size $13 \times 13 \times 13$) contains approximately 26,000 voxels, or 2.94% of the same size. These results suggest that the proposed pipeline allows processing of only a fraction of the input volume, with an additional memory overhead below 1 GB and extra execution time for sampling and reconstruction under 0.5 s. This shows that the proposed sampling pipeline can be used in environments with limited resources, addressing RQ3 by demonstrating that memory usage and inference time remain manageable.

5.4 Ablation Study

In this section, we present an ablation study to systematically evaluate the impact of various hyperparameter choices on the performance of the sampling pipeline. This

includes modifications to the training loss, the size of the projection neighborhood k, and the number of latent vector optimization steps T used during reconstruction. We present the ablation results using a patch configuration of 4 patches of size $13 \times 13 \times 13$ on the VerSe [SHB+21] dataset. The results on the selected patch configuration and the dataset generalize well across all other evaluated patch configurations and datasets. The modifications to the training loss and the projection neighborhood size k are independent of the patch size and scale with the number of sampled patches. We select the VerSe [SHB+21] dataset due to its more complex geometry and higher inter-instance variability compared to Total Segmentator Kindeys [WBM+23].

5.4.1 Training Loss

The proposed sampling pipeline is optimized using a multi-term loss based on Lang et al. [LMA20]. However, due to the differences in the input point cloud characteristics and the subsequent processing of the sampled 3D points between Lang et al. [LMA20] and the proposed sampling pipeline, we adapt the original training loss (details in Section 3.6). To assess the impact of proposed loss adjustments, we evaluate each modified term and compare it against the original loss formulation proposed by Lang et al. [LMA20].

Proximity Loss The proximity loss encourages the Samples Proposal Stage f_{gen} of the Point Cloud Sampling component (Section 3.3) to generate points $g \in G$ closer to points p of the input point cloud P. This work follows Lang et al. [LMA20] and uses the average nearest-neighbor distance loss, defined in Equation 3.13, as the proximity loss. However, in Lang et al. [LMA20], the proximity loss is defined as the sum of the average nearest-neighbor distance loss and an additional max proximity term, defined as:

$$\mathcal{L}_{\text{max\_proximity}}(G, P) = \max_{\mathbf{g} \in G} \min_{\mathbf{p} \in P} \|\mathbf{g} - \mathbf{p}\|_2^2. \tag{5.1}$$

We assess the impact of the max-proximity loss $\mathcal{L}_{\text{max\_proximity}}$ on the sampling pipeline by comparing models trained with and without this term under a configuration of 4 patches of size $13 \times 13 \times 13$. Evaluation results, summarized in Table 5.10, demonstrate no significant performance differences between the two models. As a result, we do not include the max proximity loss term in the sampling loss formulation.

Repulsion Loss The purpose of the repulsion loss is to encourage a spread of the projected points $r \in \mathcal{R}$ to achieve better exploration of the input point cloud. In Lang et al. [LMA20], the repulsion loss is defined as:

$$\mathcal{L}^{\text{orig}}_{\text{repulsion}}(G, P) = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \min_{\mathbf{g} \in G} \|\mathbf{p} - \mathbf{g}\|_2^2. \tag{5.2}$$

This definition of the repulsion loss is suitable for point clouds sampled from a 2D manifold embedded in $\mathbb{R}^3$, obtained by, e.g., sampling a 3D mesh at discrete locations, as

Proximity Loss	DICE↑	ASD↓	HD_{95} ↓
$\mathcal{L}_{\text{proximity}}$ (Proposed)	$\mathbf{0.800 \pm 0.07}$	$\mathbf{1.385 \pm 0.59}$	$\mathbf{4.545 \pm 2.20}$
$\mathcal{L}_{\text{proximity}} + \mathcal{L}_{\text{max\_proximity}}$ (Lang et al. [LMA20])	0.797 ± 0.06	1.402 ± 0.63	4.639 ± 2.32

Table 5.10: Comparison of evaluation metrics for proximity loss formulations using only $\mathcal{L}_{\text{proximity}}$ (proposed) and the combined formulation $\mathcal{L}_{\text{proximity}} + \mathcal{L}_{\text{max\_proximity}}$ as used by Lang et al. [LMA20].

Repulsion Implementation	DICE↑	ASD↓	HD_{95} ↓
Proposed (Equation 3.14)	$\mathbf{0.800 \pm 0.07}$	$\mathbf{1.385 \pm 0.59}$	$\mathbf{4.545 \pm 2.20}$
Lang et al. [LMA20] (Equation 5.2)	0.776 ± 0.06	1.594 ± 0.61	5.403 ± 2.51

Table 5.11: Comparison of evaluation metrics for the proposed implementation of the repulsion loss and the original implementation introduced by Lang et al. [LMA20].

is the case of the dataset used by Lang et al. [LMA20]. However, in the proposed pipeline, point clouds are generated by converting a dense voxel grid into a point representation. As a result, points may occupy arbitrary positions within the three-dimensional volume and are not constrained to lie on a surface manifold. Therefore, we employ an alternative repulsion loss formulation, introduced in Equation 3.14. To support this design choice, Table 5.11 reports a comparative evaluation of models trained using the original repulsion loss and the proposed loss formulation.

Projection Loss In Lang et al. [LMA20], the overall sampler training loss is additionally extended by a projection loss, defined as $\mathcal{L}_{\text{proj}} = t^2$, where t is the temperature parameter from the soft-projection module (Equation 3.3). As explained in Section 3.6, having the temperature parameter in the loss encourages it to approach zero, which leads to nearest neighbor approximation in the soft-projection module [LMA20]. In our pipeline, we keep the temperature parameter t trainable but do not encourage changes in its value through any loss. We argue that projecting generated points using approximate nearest-neighbor matching does not support our goals, as we aim to obtain patch centers within the simplified point cloud. Using nearest-neighbor matching restricts patch center selection to points already present in the simplified cloud and prevents the extraction of points that lie between them. We support this argument by comparing training runs with and without the temperature t included as a loss term using a configuration of 4 patches of size $13 \times 13 \times 13$, and we report the resulting metrics in Table 5.12.

Parameter t	**DICE↑**	**ASD↓**	**HD_{95} ↓**
Not Included in the loss (Proposed)	$\mathbf{0.800 \pm 0.07}$	$\mathbf{1.385 \pm 0.59}$	$\mathbf{4.545 \pm 2.20}$
Included in the loss (Lang et al. [LMA20])	0.788 ± 0.06	1.478 ± 0.53	4.910 ± 2.21

Table 5.12: Comparison of evaluation metrics for configurations using the proposed loss formulation without the parameter t and the original formulation including t as used in Lang et al. [LMA20].

Parameter k	**DICE↑**	**ASD↓**	**HD_{95} ↓**
32 (Proposed)	$\mathbf{0.800 \pm 0.07}$	$\mathbf{1.385 \pm 0.59}$	$\mathbf{4.545 \pm 2.20}$
16 (Lang et al. [LMA20])	0.791 ± 0.07	1.412 ± 0.60	4.735 ± 2.31

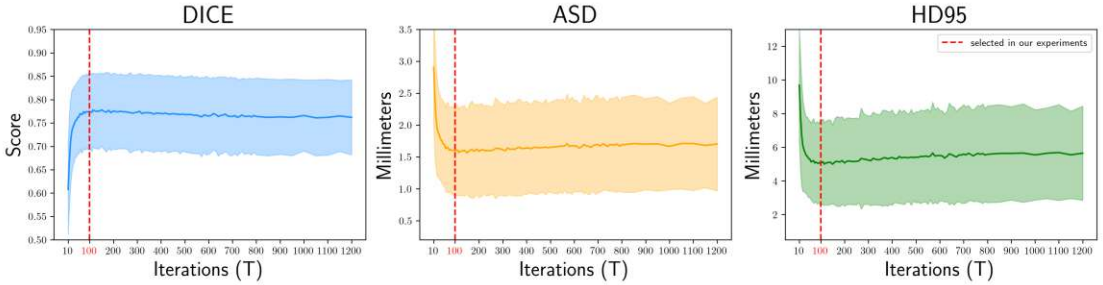
Table 5.13: Comparison of evaluation metrics for configurations using the proposed value of k and the value adopted by Lang et al. [LMA20].

5.4.2 Projection Neighborhood Size k

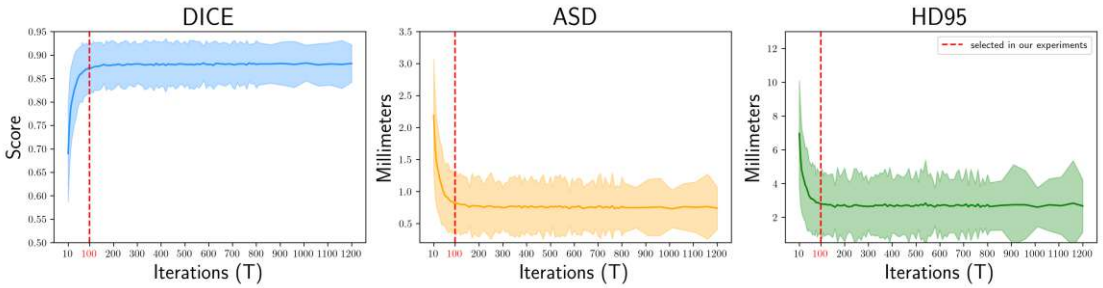
The projection neighborhood size k controls the number of neighbors onto which each generated point is projected during the soft projection operation. A small neighborhood size limits point cloud exploration by restricting the receptive field, whereas an excessive size may lead to the loss of local context [LMA20]. In the original SampleNet implementation, Lang et al. [LMA20] select $k = 16$ for all reconstruction experiments, while in our implementation, we empirically found $k = 32$ to work better. Table 5.13 compares reconstruction metrics for $k = \{16, 32\}$, supporting our choice of the parameter k.

5.4.3 Number of Latent Vector Optimization Steps T

In our work, the reconstruction network generates the full-resolution reconstruction by evaluating coordinates along with a latent vector z. The latent vector z encodes a shape-specific representation, allowing a single model to capture a family of shapes sharing a common prior. In order to find a suitable latent vector representing the shape currently being reconstructed, a randomly initialized latent vector z is optimized by predicting occupancy values at coordinates belonging to each sampled patch (Equation 3.10 and Figure 3.5). The optimization is performed in a loop using T steps of the gradient descent algorithm, with the choice of T influencing the accuracy of the reconstruction as well as the reconstruction speed and memory required [ALL+22]. In the original implementation, Amiranashvili et al. [ALL+22] optimize the latent vector using $T = 1200$ steps for the VerSe [SHB+21] dataset. In all our experiments, we select $T = 100$. This choice is based on the hypothesis that fewer optimization steps may still yield comparable performance while reducing computational overhead. To investigate this, we applied the baseline FPS



(a) 4 patches of size $13 \times 13 \times 13$.



(b) 12 patches of size $13 \times 13 \times 13$.

Figure 5.17: Reconstruction performance evaluated for different numbers of latent vector optimization steps T. The red line represents $T = 100$, which is further utilized in all our experiments.

algorithm to each sample in the VerSe dataset [SHB+21] to obtain a set of volumetric patches, and then evaluated how well the reconstruction network reconstructs from these patches for varying values of T. Figure 5.17 illustrates how the number of optimization steps impacts the performance of the reconstruction model. Figure 5.17a hints that with smaller context (4 patches of size $13 \times 13 \times 13$) the reconstruction network might overfit for large T, while Figure 5.17b illustrates that with larger context (12 patches of size $13 \times 13 \times 13$) the reconstruction network's performance plateaus for large T. For this reason, we selected $T = 100$ as the preferred trade-off between reconstruction performance and computational demands.

5.5 Summary of Key Results

We summarize the key finding of the thesis as:

- The sampled patches identified by the proposed end-to-end pipeline achieve reconstruction DICE scores of 0.758–0.862 on VerSe [SHB+21] and 0.842–0.924 on TotalSegmentator Kidneys [WBM+23], depending on the selected patch configuration.

- The learned sampling component outperforms selected non-learned baselines in cases where patch configurations are small, the sampled object has complex geometry, or the performance of the reconstruction network is degraded. In these cases, learned sampling improves the reconstruction DICE scores by 1–5% for VerSe [SHB+21] and 0–1% for TotalSegmentator Kidneys [WBM+23].
- The learned sampling component is capable of selecting patches from the entire volume, including interior regions, while baseline methods only cover surface areas and fail to capture internal structures.
- The sampling and reconstruction networks introduce less than 0.5 s of additional computational overhead and require under 1 GB of extra memory.

5.6 Limitations

The biggest limitation of the proposed pipeline is its computational complexity during training, both in terms of time and memory. The overall memory and computational demands are primarily influenced by two factors: patch configuration (number of patches and patch size) and the spatial resolution of the input voxel grid. Larger patches or a higher number of patches increase the number of output voxels extracted by the pipeline, which directly affects the reconstruction. With more extracted patch voxels, the optimization of the shape-specific latent vector z requires storing gradients with respect to all these voxels, resulting in longer optimization steps and substantially higher memory usage during gradient backpropagation. In a similar manner, when the input voxel grid has higher spatial resolution, the total number of voxels for reconstruction also increases, which, as a result, leads to larger dense sampling grid Ω. During the forward pass, gradients must be stored for every voxel in Ω, which directly increases memory demands and computational time.

Another limitation is the fixed number of points n used during voxel-to-point cloud conversion. In all our experiments, we set $n = 2048$, which is consistent with the literature and allows us to use existing sampling architectures without modifying the number of layers or their sizes, as these architectures were benchmarked with point clouds containing 2048 points. In our pipeline, we convert input voxels grids to point clouds and downsample to $n = 2048$, which creates a lightweight shape approximation. While the approximations obtained with $n = 2048$ provide enough context and details when sampling voxel grids containing tens of thousands occupied voxels, it is unclear, whether this number is sufficient for voxel grids with millions or more occupied voxels.

CHAPTER 6

Conclusion and Future Work

This thesis presents a deep learning-based approach for informed patch sampling from medical shape segmentations. The learned sampling component identifies the most informative sub-volumes from an input voxel grid, selecting patches that contribute most effectively to accurate reconstruction. We review existing learned sampling methods primarily applied in the point cloud domain and adapt a suitable method for the volumetric data domain. We combine the learned sampler with a reconstruction network based on an implicit neural representation into an end-to-end pipeline that accelerates the processing of large volumetric scans by selecting a subset of the input medical scan for processing and then reconstructing the remaining values from the processed subset. This can significantly reduce computational effort and memory requirements, enabling deployment in resource-constrained settings where processing the full volume would be infeasible.

6.1 Summary and Answers to Research Questions

In our proposed sampling pipeline, we convert the binary segmentation represented as a voxel grid into a point cloud by treating each occupied voxel as a point, while keeping the voxel grid aside for patch extraction. To obtain point clouds of a consistent size and produce a simplified shape approximation, we employ Sobol downsampling. A point-cloud–based sampling architecture, SampleNet, adapted from Lang et al. [LMA20], then processes the simplified point cloud and generates a set of samples, representing the centers of volumetric patches for sampling. These samples, when used as centers of patches, may not be positioned sufficiently close to the input point cloud to ensure an intersection with the underlying voxel grid. As a result, such extracted patches will not contain any information about the shape reconstructed. To address this, we use the Soft Projection module from SampleNet, which projects each generated point onto its nearest neighbors in the input point cloud and represents the projection as a

weighted sum of these neighbors. A patch extraction module then treats the projected points as centers and extracts volumetric patches from the original voxel grid. These patches are subsequently fed into the neural reconstruction network from Amiranashvili et al. [ALL+22], which produces the full-resolution reconstruction.

We optimize our pipeline using a multi-term loss function. The proximity loss encourages generated points to remain close to the input point cloud. The repulsion loss prevents patch centers from collapsing into one another, producing overlapping patches. Finally, the reconstruction loss, composed of per-voxel binary cross-entropy and the soft DICE score, provides gradients from the reconstruction network, ensuring that the sampling operation is optimized for the reconstruction task.

To address **RQ1**, we evaluate the ability of the learned sampling component to identify the most informative patches for volumetric reconstruction. We conduct experiments on two datasets of differing shape complexity: kidneys (simple shape) and vertebrae (complex shape), and further assess the influence of reconstruction network quality on selected patches. Results show that the sampler effectively selects areas that contribute most to the reconstruction accuracy. For simpler shapes, the learned sampler achieves performance comparable to baseline methods when using a high-accuracy reconstruction network. However, our results suggest that, with a low-quality reconstruction model or when sampling complex shapes, the learned sampler consistently selects regions that improve reconstruction performance.

To address **RQ2**, we investigate how patch size and the number of patches influence reconstruction quality. To this end, we evaluate multiple patch configurations, varying both the number and size of patches. In many cases, using more of smaller patches yields better results than using fewer larger ones, especially for the vertebra dataset. Furthermore, we show that when the number of patches or their size is small, the learned sampler significantly outperforms baseline methods. This highlights the superiority of the learned sampler when the available input is very limited.

Finally, to address **RQ3**, we analyze the computational and memory demands of the proposed sampling pipeline across different patch sizes and numbers of sampled patches. Our results show that even for the largest evaluated patch configurations, the additional memory overhead of the end-to-end pipeline remains below 1 GB and the extra computation remains under 0.5 s. This demonstrates that the proposed sampling pipeline can be deployed in resource-constrained environments, potentially speeding up downstream processing or enabling it in scenarios where full-volume processing was previously infeasible due to memory limitations.

6.2 Future Work

The main limitation of the current pipeline is its long training time and high memory requirements. This is primarily caused by the need to backpropagate through all optimization steps of the latent vector in the reconstruction network, followed by the

subsequent dense, full-resolution inference. Reducing training time and memory requirements would make the pipeline more practical for large-scale medical datasets, enabling evaluation of larger patch configurations. One way to address this limitation is to leverage reconstruction networks that achieve accurate results while consuming less memory and incurring lower computational cost. As an example, we highlight the work of De Paolis et al. [DPLN+24], who demonstrate that combining implicit neural representation with meta-learning achieves high reconstruction accuracy after a single optimization step. Another limitation of the proposed pipeline is that the learned sampling strategy is tightly coupled with the downstream task network. This dependency limits the applicability of the approach in scenarios where such a task network is unavailable or where multiple downstream tasks are considered. In future work, this limitation could be addressed by adopting a sampling architecture that does not require a task network for training. For example, Yang et al. [YSA23] propose a self-supervised sampling network that learns to order the points within a point cloud using hierarchical contrastive loss. Lastly, another possibility is to explore increasing the number n of points used during voxel-to-point cloud downsampling. In our experiments, we are using $n = 2048$, which we successfully applied to voxel grids containing fewer than 100,000 occupied voxels. However, it remains unclear whether this number is sufficient for voxel grids with a larger number of occupied voxels. Increasing n could provide finer shape approximations, leading to improved reconstruction quality.

Overview of Generative AI Tools Used

Grammarly Grammarly was used to identify spelling errors, missing punctuation, and basic grammar issues.

ChatGPT ChatGPT was used only as an initial drafting tool. A list of original arguments, thoughts, opinions, and ideas was provided. The model generated a first draft version of the paragraph, which was then substantially manually revised. During the revision, the text was edited to fully reflect the original thoughts, with only minor elements, such as sentence structure and conjunctions, occasionally kept from the original draft.

List of Figures

List of Tables

Bibliography

[AB11] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE11, page 1–10. ACM, May 2011.

[AHL+23] Arman Avesta, Sajid Hossain, MingDe Lin, Mariam Aboian, Harlan M. Krumholz, and Sanjay Aneja. Comparing 3D, 2.5D, and 2D Approaches to Brain Image Auto-Segmentation. *Bioengineering*, 10(2):181, February 2023.

[ALL+22] Tamaz Amiranashvili, David Lüdke, Hongwei Bran Li, Bjoern Menze, and Stefan Zachow. Learning Shape Reconstruction from Sparse Measurements with Neural Implicit Functions. In Ender Konukoglu, Bjoern Menze, Archana Venkataraman, Christian Baumgartner, Qi Dou, and Shadi Albarqouni, editors, *Proceedings of The 5th International Conference on Medical Imaging with Deep Learning*, volume 172 of *Proceedings of Machine Learning Research*, pages 22–34. PMLR, 06–08 Jul 2022.

[ALL+24] Tamaz Amiranashvili, David Lüdke, Hongwei Bran Li, Stefan Zachow, and Bjoern H Menze. Learning continuous shape priors from sparse data with neural implicit functions. *Medical Image Analysis*, 94:103099, May 2024.

[AÖ17] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, November 2017.

[BJLPS18] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the Latent Space of Generative Networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 600–609. PMLR, 10–15 Jul 2018.

[BSB+17] Sara Broggi, Elisa Scalco, Maria Luisa Belli, Gerlinde Logghe, Dirk Verellen, Stefano Moriconi, Anna Chiara, Anna Palmisano, Renata

Mellone, Claudio Fiorino, and Giovanna Rizzo. A Comparative Evaluation of 3 Different Free-Form Deformable Image Registration and Contour Propagation Methods for Head and Neck MRI: The Case of Parotid Changes During Radiotherapy. *Technology in Cancer Research and Treatment*, 16(3):373–381, February 2017.

[BTKA18] Stefano B. Blumberg, Ryutaro Tanno, Iasonas Kokkinos, and Daniel C. Alexander. *Deeper Image Quality Transfer: Training Low-Memory Neural Networks for 3D Images*, page 118–125. Springer International Publishing, 2018.

[BTZ+21] Miguel Angel Bautista, Walter Talbott, Shuangfei Zhai, Nitish Srivastava, and Joshua M. Susskind. On the generalization of learning-based 3D reconstruction. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, January 2021.

[CAN87] JOHN CANNY. *A Computational Approach to Edge Detection*, page 184–203. Elsevier, 1987.

[CCQ+26] Xiaolei Chen, Jie Chen, Shoumeng Qiu, Xiangyang Xue, and Jian Pu. GS-Net: Point cloud sampling with graph neural networks. *Pattern Recognition*, 170:112054, February 2026.

[CET01] Timothy F. Cootes, Gareth J. Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.

[CH67] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[CKZ+24] Hongbo Chen, Logiraj Kumaralingam, Shuhang Zhang, Sheng Song, Fayi Zhang, Haibin Zhang, Thanh-Tu Pham, Kumaradevan Punithakumar, Edmond H.M. Lou, Yuyao Zhang, Lawrence H. Le, and Rui Zheng. Neural implicit surface reconstruction of freehand 3D ultrasound volume with geometric constraints. *Medical Image Analysis*, 98:103305, December 2024.

[CML+24] Jieneng Chen, Jieru Mei, Xianhang Li, Yongyi Lu, Qihang Yu, Qingyue Wei, Xiangde Luo, Yutong Xie, Ehsan Adeli, Yan Wang, Matthew P. Lungren, Shaoting Zhang, Lei Xing, Le Lu, Alan Yuille, and Yuyin Zhou. TransUNet: Rethinking the U-Net architecture design for medical image segmentation through the lens of transformers. *Medical Image Analysis*, 97:103280, October 2024.

[CSKG17] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.

In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 77–85. IEEE, July 2017.

[CTV19] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable Ranking and Sorting using Optimal Transport. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[CZ19] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 5932–5941. IEEE, June 2019.

[DGBCNAVA17] Jesus Diaz Garcia, Pere Brunet Crosa, Isabel Navazo Alvaro, and Pere Pau Vazquez Alcocer. Downsampling methods for medical datasets. In *Proceedings of the International conferences Computer Graphics, Visualization, Computer Vision and Image Processing 2017 and Big Data Analytics, Data Mining and Computational Intelligence 2017: Lisbon, Portugal, July 21-23, 2017*, pages 12–20. IADIS Press, 2017.

[DLA19] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to Sample. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 2755–2764. IEEE, June 2019.

[DPLN+24] Gaia Romana De Paolis, Dimitrios Lenis, Johannes Novotny, Maria Wimmer, Astrid Berg, Theresa Neubauer, Philip Matthias Winter, David Major, Ariharasudhan Muthusami, Gerald Schröcker, Martin Mienkina, and Katja Bühler. *Fast Medical Shape Reconstruction via Meta-learned Implicit Neural Representations*, page 189–204. Springer Nature Switzerland, October 2024.

[DYY+17] Bryson Dietz, Eugene Yip, Jihyun Yun, B. Gino Fallone, and Keith Wachowicz. Real-time dynamic <scp>MR</scp> image reconstruction using compressed sensing and principal component analysis (<scp>CS</scp>-<scp>PCA</scp>): Demonstration in lung tumor tracking. *Medical Physics*, 44(8):3978–3989, June 2017.

[ELPZ97] Y. Eldar, M. Lindenbaum, M. Porat, and Y.Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, September 1997.

[GAY+19] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized Inner Loop Meta-Learning. *arXiv e-prints*, page arXiv:1910.01727, October 2019.

[GLB+19] Sambit Ghadai, Xian Yeow Lee, Aditya Balu, Soumik Sarkar, and Adarsh Krishnamurthy. Multi-Level 3D CNN for Learning Multi-Scale Spatial Features. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 1152–1156. IEEE, June 2019.

[GTQ+18] Andrew T. Grainger, Nicholas J. Tustison, Kun Qing, Rene Roy, Stuart S. Berr, and Weibin Shi. Deep learning-based quantification of abdominal fat on magnetic resonance images. *PLOS ONE*, 13(9):e0204071, September 2018.

[HDNK24] David Jozef Hresko, Peter Drotar, Quoc Cuong Ngo, and Dinesh Kant Kumar. Enhanced Domain Adaptation for Foot Ulcer Segmentation Through Mixing Self-Trained Weak Labels. *Journal of Imaging Informatics in Medicine*, 38(1):455–466, July 2024.

[HHGCO20] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. PointGMM: A Neural GMM Network for Point Clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 12051–12060. IEEE, June 2020.

[Ho,21] Ho, Ngoc-Vuong and Nguyen, Tan and Diep, Gia-Han and Le, Ngan and Hua, Binh-Son. Point-Unet: A Context-Aware Point-Based Neural Network for Volumetric Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 644–655. Springer, 2021.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[IWU+24] Fabian Isensee, Tassilo Wald, Constantin Ulrich, Michael Baumgartner, Saikat Roy, Klaus Maier-Hein, and Paul F. Jäger. *nnU-Net Revisited: A Call for Rigorous Validation in 3D Medical Image Segmentation*, page 488–498. Springer Nature Switzerland, 2024.

[JMFU17] Kyong Hwan Jin, Michael T. McCann, Emmanuel Froustey, and Michael Unser. Deep Convolutional Neural Network for Inverse

Problems in Imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, September 2017.

[JYF+25] Young Seok Jeon, Hongfei Yang, Huazhu Fu, Yeshe Kway, and Mengling Feng. *No More Sliding Window: Efficient 3D Medical Image Segmentation with Differentiable Top-K Patch Sampling*, page 376–386. Springer Nature Switzerland, September 2025.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 12 2014.

[KBDB+19] Hugo J. Kuijf, J. Matthijs Biesbroek, Jeroen De Bresser, Rutger Heinen, Simon Andermatt, Mariana Bento, Matt Berseth, Mikhail Belyaev, M. Jorge Cardoso, Adrià Casamitjana, D. Louis Collins, Mahsa Dadar, Achilleas Georgiou, Mohsen Ghafoorian, Dakai Jin, April Khademi, Jesse Knight, Hongwei Li, Xavier Lladó, Miguel Luna, Qaiser Mahmood, Richard McKinley, Alireza Mehrtash, Sébastien Ourselin, Bo-Yong Park, Hyunjin Park, Sang Hyun Park, Simon Pezold, Elodie Puybareau, Leticia Rittner, Carole H. Sudre, Sergi Valverde, Verónica Vilaplana, Roland Wiest, Yongchao Xu, Ziyue Xu, Guodong Zeng, Jianguo Zhang, Guoyan Zheng, Christopher Chen, Wiesje van der Flier, Frederik Barkhof, Max A. Viergever, and Geert Jan Biessels. Standardized Assessment of Automatic Segmentation of White Matter Hyperintensities and Results of the WMH Segmentation Challenge. *IEEE Transactions on Medical Imaging*, 38(11):2556–2568, 2019.

[KCDS11] Florian Knoll, Christian Clason, Clemens Diwoky, and Rudolf Stollberger. Adapted random sampling patterns for accelerated MRI. *Magnetic Resonance Materials in Physics, Biology and Medicine*, 24(1):43–50, January 2011.

[KJI+24] Amarjeet Kumar, Hongxu Jiang, Muhammad Imran, Cyndi Valdes, Gabriela Leon, Dahyun Kang, Parvathi Nataraj, Yuyin Zhou, Michael D. Weiss, and Wei Shao. A flexible 2.5D medical image segmentation approach with in-slice and cross-slice attention. *Computers in Biology and Medicine*, 182:109173, November 2024.

[KMY17] Eunhee Kang, Junhong Min, and Jong Chul Ye. A deep convolutional neural network using directional wavelets for low-dose X-ray CT reconstruction. *Medical Physics*, 44(10), October 2017.

[LJZ+18] Hongwei Li, Gongfa Jiang, Jianguo Zhang, Ruixuan Wang, Zhaolei Wang, Wei-Shi Zheng, and Bjoern Menze. Fully convolutional network ensembles for white matter hyperintensities segmentation in MR images. *NeuroImage*, 183:650–665, December 2018.

[LLG+25] Zhuangzi Li, Shan Liu, Wei Gao, Guanbin Li, and Ge Li. S4R: Rethinking Point Cloud Sampling via Guiding Upsampling-Aware Perception. *IEEE Transactions on Multimedia*, 27:6677–6689, 2025.

[LLX+22] Xiangde Luo, Wenjun Liao, Jianghong Xiao, Jieneng Chen, Tao Song, Xiaofan Zhang, Kang Li, Dimitris N. Metaxas, Guotai Wang, and Shaoting Zhang. WORD: A large scale dataset, benchmark and clinical applicable study for abdominal organ segmentation from CT image. *Medical Image Analysis*, 82:102642, November 2022.

[LLZ21] Chenlei Lv, Weisi Lin, and Baoquan Zhao. Approximate Intrinsic Voxel Structure for Point Cloud Simplification. *IEEE Transactions on Image Processing*, 30:7241–7255, 2021.

[LMA20] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable Point Cloud Sampling. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 7575–7585. IEEE, June 2020.

[LMU21] Jörn Lötsch, Sebastian Malkusch, and Alfred Ultsch. Optimal distribution-preserving downsampling of large biomedical data sets (opdisDownsampling). *PLOS ONE*, 16(8):e0255838, August 2021.

[LTLH19] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-Voxel CNN for Efficient 3D Deep Learning. In *Advances in Neural Information Processing Systems*, 2019.

[LTLS21] Jaeho Lee, Jihoon Tack, Namhoon Lee, and Jinwoo Shin. Meta-learning Sparse Implicit Neural Representations. In *Advances in Neural Information Processing Systems*, 2021.

[LWW+11] Carsten Last, Simon Winkelbach, Friedrich M. Wahl, Klaus W. G. Eichhorn, and Friedrich Bootz. *A Locally Deformable Statistical Shape Model*, page 51–58. Springer Berlin Heidelberg, 2011.

[LXZ+18] Siqi Liu, Daguang Xu, S. Kevin Zhou, Olivier Pauly, Sasa Grbic, Thomas Mertelmeier, Julia Wicklein, Anna Jerebko, Weidong Cai, and Dorin Comaniciu. *3D Anisotropic Hybrid Network: Transferring Convolutional Features from 2D Images to 3D Anisotropic Volumes*, page 851–858. Springer International Publishing, 2018.

[LYY17] Dongwook Lee, Jaejun Yoo, and Jong Chul Ye. Deep residual learning for compressed sensing MRI. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, page 15–18. IEEE, April 2017.

[MAT+23] Amirali Molaei, Amirhossein Aminimehr, Armin Tavakoli, Amirhossein Kazerouni, Bobby Azad, Reza Azad, and Dorit Merhof. Implicit Neural Representation in Medical Imaging: A Comparative Survey. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, page 2373–2383. IEEE, October 2023.

[MBL+02] S.C. Mitchell, J.G. Bosch, B.P.F. Lelieveldt, R.J. van der Geest, J.H.C. Reiber, and M. Sonka. 3-D active appearance models: segmentation of cardiac MR and ultrasound images. *IEEE Transactions on Medical Imaging*, 21(9):1167–1178, September 2002.

[MC24] Charles Millard and Mark Chiew. Clean self-supervised MRI reconstruction from noisy, sub-sampled training data with Robust SSDU. *Bioengineering*, 11(12):1305, 2024.

[MCAL16] Awais Mansoor, Juan J. Cerrolaza, Robert A. Avery, and Marius G. Linguraru. *Partitioned Shape Modeling with On-the-Fly Sparse Appearance Learning for Anterior Visual Pathway Segmentation*, page 104–112. Springer International Publishing, 2016.

[MD03] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003.

[MON+19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 4455–4465. IEEE, June 2019.

[MTMHS21] Javier Montalt-Tordera, Vivek Muthurangu, Andreas Hauptmann, and Jennifer Anne Steeden. Machine learning in Magnetic Resonance Imaging: Image reconstruction. *Physica Medica*, 83:79–87, March 2021.

[MWL+17] Jingting Ma, Anqi Wang, Feng Lin, Stefan Wesarg, and Marius Erdt. *Nonlinear Statistical Shape Modeling for Ankle Bone Segmentation Using a Novel Kernelized Robust PCA*, page 136–143. Springer International Publishing, 2017.

[MYK+25] Jun Ma, Zongxin Yang, Sumin Kim, Bihui Chen, Mohammed Baharoon, Adibvafa Fallahpour, Reza Asakereh, Hongwei Lyu, and Bo Wang. MedSAM2: Segment Anything in 3D Medical Images and Videos. *arXiv preprint arXiv:2504.03600*, 2025.

[NMW+19] Alexey A. Novikov, David Major, Maria Wimmer, Dimitrios Lenis, and Katja Buhler. Deep Sequential Segmentation of Organs in Volumetric Medical Scans. *IEEE Transactions on Medical Imaging*, 38(5):1207–1215, May 2019.

[OB22] Amine Ouasfi and Adnane Boukhayma. *Few 'Zero Level Set'-Shot Learning of Shape Signed Distance Functions in Feature Space*, page 561–578. Springer Nature Switzerland, 2022.

[PFS+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 165–174. IEEE, June 2019.

[PHH21] Daniel Patel, Thomas Höllt, and Markus Hadwiger. *Real-Time Algorithms for Visualizing and Processing Seismic and Reservoir Data*, page 45–78. Springer International Publishing, 2021.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, page 234–241. Springer International Publishing, 2015.

[RJC+12] Pablo Ruisoto, Juan Antonio Juanes, Israel Contador, Paula Mayoral, and Alberto Prats-Galino. Experimental evidence for improved neuroimaging interpretation using three-dimensional graphic models. *Anatomical Sciences Education*, 5(3):132–137, March 2012.

[RTH+22] Anish Raj, Fabian Tollens, Laura Hansen, Alena-Kathrin Golla, Lothar R. Schad, Dominik Nörenberg, and Frank G. Zöllner. Deep Learning-Based Total Kidney Volume Segmentation in Autosomal Dominant Polycystic Kidney Disease Using Attention, Cosine Loss, and Sharpness Aware Minimization. *Diagnostics*, 12(5):1159, May 2022.

[SAG84] T.W. Sederberg, D.C. Anderson, and R.N. Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics, and Image Processing*, 27(2):247, August 1984.

[SCH+18] Jo Schlemper, Jose Caballero, Joseph V. Hajnal, Anthony N. Price, and Daniel Rueckert. A Deep Cascade of Convolutional Neural Networks for Dynamic MR Image Reconstruction. *IEEE Transactions on Medical Imaging*, 37(2):491–503, February 2018.

[SCT+20] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-Learning Signed Distance Functions. In *Proc. NeurIPS*, 2020.

[SEJ18] Johan Swärd, Filip Elvander, and Andreas Jakobsson. Designing sampling schemes for multi-dimensional data. *Signal Processing*, 150:1–10, September 2018.

[SGT+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[SHB+21] Anjany Sekuboyina, Malek E. Husseini, Amirhossein Bayat, Maximilian Löffler, Hans Liebl, Hongwei Li, Giles Tetteh, Jan Kukačka, Christian Payer, Darko Štern, Martin Urschler, Maodong Chen, Dalong Cheng, Nikolas Lessmann, Yujin Hu, Tianfu Wang, Dong Yang, Daguang Xu, Felix Ambellan, Tamaz Amiranashvili, Moritz Ehlke, Hans Lamecker, Sebastian Lehnert, Marilia Lirio, Nicolás Pérez de Olaguer, Heiko Ramm, Manish Sahu, Alexander Tack, Stefan Zachow, Tao Jiang, Xinjun Ma, Christoph Angerman, Xin Wang, Kevin Brown, Alexandre Kirszenberg, Élodie Puybareau, Di Chen, Yiwei Bai, Brandon H. Rapazzo, Timyoas Yeah, Amber Zhang, Shangliang Xu, Feng Hou, Zhiqiang He, Chan Zeng, Zheng Xiangshang, Xu Liming, Tucker J. Netherton, Raymond P. Mumme, Laurence E. Court, Zixun Huang, Chenhang He, Li-Wen Wang, Sai Ho Ling, Lê Duy Huỳnh, Nicolas Boutry, Roman Jakubicek, Jiri Chmelik, Supriti Mulay, Mohanasankar Sivaprakasam, Johannes C. Paetzold, Suprosanna Shit, Ivan Ezhov, Benedikt Wiestler, Ben Glocker, Alexander Valentinitsch, Markus Rempfler, Björn H. Menze, and Jan S. Kirschke. VerSe: A Vertebrae labelling and segmentation benchmark for multi-detector CT images. *Medical Image Analysis*, 73:102166, October 2021.

[SMB+20] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *arXiv*, 2020.

[Sob76] I.M. Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, January 1976.

[SPX24] Liyue Shen, John Pauly, and Lei Xing. NeRP: Implicit Neural Representation Learning With Prior Embedding for Sparsely Sampled Image Reconstruction. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):770–782, January 2024.

[Stu08] Student. The Probable Error of a Mean. *Biometrika*, 6(1):1, March 1908.

[TBO+21] Federico Turella, Gustav Bredell, Alexander Okupnik, Sebastiano Caprara, Dimitri Graf, Reto Sutter, and Ender Konukoglu. *High-Resolution Segmentation of Lumbar Vertebrae from Conventional Thick Slice MRI*, page 689–698. Springer International Publishing, 2021.

[TKMSH23] Athanasios Tragakis, Chaitanya Kaul, Roderick Murray-Smith, and Dirk Husmeier. The Fully Convolutional Transformer for Medical Image Segmentation. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, page 3649–3658. IEEE, January 2023.

[TTKORS20] Alžběta Turečková, Tomáš Tureček, Zuzana Komínková Oplatková, and Antonio Rodríguez-Sánchez. Improving CT Image Tumor Segmentation Through Deep Supervision and Attentional Gates. *Frontiers in Robotics and AI*, 7, August 2020.

[TTM+22] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Nießner, J. T. Barron, G. Wetzstein, M. Zollhöfer, and V. Golyanik. Advances in Neural Rendering. *Computer Graphics Forum*, 41(2):703–735, 2022.

[VCA+17] Benjamin Villard, Valentina Carapella, Rina Ariga, Vicente Grau, and Ernesto Zacur. *Cardiac Mesh Reconstruction from Sparse, Heterogeneous Contours*, page 169–181. Springer International Publishing, 2017.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wan24] Zhengren Wang. 3d representation methods: A survey. *CoRR*, abs/2410.06475, 2024.

[WBM+23] Jakob Wasserthal, Hanns-Christian Breit, Manfred T. Meyer, Maurice Pradella, Daniel Hinck, Alexander W. Sauter, Tobias Heye, Daniel T. Boll, Joshy Cyriac, Shan Yang, Michael Bach, and Martin Segeroth. TotalSegmentator: Robust Segmentation of 104 Anatomic Structures in CT Images. *Radiology: Artificial Intelligence*, 5(5), September 2023.

[WCW+22] Wenxuan Wang, Chen Chen, Jing Wang, Sen Zha, Yan Zhang, and Jiangyun Li. *Med-DANet: Dynamic Architecture Network for Efficient Medical Volumetric Segmentation*, page 506–522. Springer Nature Switzerland, 2022.

[WF06] Oliver Williams and Andrew Fitzgibbon. Gaussian process implicit surfaces. In *Gaussian Processes in Practice*, 2006.

[Wil92] Frank Wilcoxon. *Individual Comparisons by Ranking Methods*, page 196–202. Springer New York, 1992.

[WLX+21] Qing Wu, Yuwei Li, Lan Xu, Ruiming Feng, Hongjiang Wei, Qing Yang, Boliang Yu, Xiaozhao Liu, Jingyi Yu, and Yuyao Zhang. *IREM: High-Resolution Magnetic Resonance Image Reconstruction via Implicit Neural Representation*, page 65–74. Springer International Publishing, 2021.

[WSL+19] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 38(5):1–12, October 2019.

[WTH+19] Yukai Wang, Qizhi Teng, Xiaohai He, Junxi Feng, and Tingrong Zhang. CT-image of rock samples super resolution using 3D convolutional neural network. *Computers and Geosciences*, 133:104314, December 2019.

[WXK+23] Jialin Wang, Nan Xiang, Navjot Kukreja, Lingyun Yu, and Hai-Ning Liang. LVDIF: a framework for real-time interaction with large volume data. *The Visual Computer*, 39(8):3373–3386, July 2023.

[WZPB23] Chengzhi Wu, Junwei Zheng, Julius Pfrommer, and Jürgen Beyerer. Attention-Based Point Cloud Edge Sampling. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 5333–5343. IEEE, June 2023.

[XDC+20] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable Top-k with Optimal Transport. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20520–20531. Curran Associates, Inc., 2020.

[XWL+24] Weiyi Xie, Nathalie Willems, Nikolas Lessmann, Tom Gibbons, and Daniele De Massari. Semi-Supervised Segmentation via Embedding Matching. In Ninon Burgos, Caroline Petitjean, Maria Vakalopoulou, Stergios Christodoulidis, Pierrick Coupe, Hervé Delingette, Carole Lartizien, and Diana Mateus, editors, *Proceedings of The 7nd International Conference on Medical Imaging with Deep Learning*, volume 250 of *Proceedings of Machine Learning Research*, pages 1741–1753. PMLR, 03–05 Jul 2024.

[XZK+26] Kangxian Xie, Yufei Zhu, Kaiming Kuang, Li Zhang, Hongwei Bran Li, Mingchen Gao, and Jiancheng Yang. Template-guided reconstruction of pulmonary segments with neural implicit functions. *Medical Image Analysis*, 109:103916, March 2026.

[YSA23] Pengwan Yang, Cees G. M. Snoek, and Yuki M. Asano. Self-Ordering Point Clouds. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, page 15767–15776. IEEE, October 2023.

[YYJ22] Yang Ye, Xiulong Yang, and Shihao Ji. APSNet: Attention Based Point Cloud Sampling. *The 33rd British Machine Vision Conference (BMVC)*, 2022.

[ZEP10] Qi Zhang, Roy Eagleson, and Terry M. Peters. Volume Visualization: A Technical Overview with a Focus on Medical Applications. *Journal of Digital Imaging*, 24(4):640–664, August 2010.

[ZRSTL18] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In Danail Stoyanov and Zeike Taylor, editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11, Cham, 2018. Springer International Publishing.