Sparse Convolutional Neural Networks for Medical Image Analysis

Jianning Li [1], Christina Gsaxner [1], Antonio Pepe [1], Dieter Schmalstieg [1], Jens Kleesiek [1], and Jan Egger [1]

[1]Affiliation not available

October 31, 2023

Abstract

Traditional convolutional neural network (CNN) methods rely on dense tensors, which makes them suboptimal for spatially sparse data. In this paper, we propose a CNN model based on sparse tensors for efficient processing of large and sparse medical images. In contrast to a dense CNN that takes the entire voxel grid as input, a sparse CNN processes only on the non-empty voxels, thus reducing the memory and computation overhead caused by the sparse input data. We evaluate our method on two clinically relevant skull reconstruction tasks: (1) given a defective skull, reconstruct the complete skull (i.e., skull shape completion), and (2) given a coarse skull, reconstruct a high-resolution skull with fine geometric details (shape super-resolution). Our method outperforms the state of the art in the skull reconstruction task quantitatively and qualitatively, while requiring substantially less memory for training and inference. We observed that, on the 3D skull data, the overall memory consumption of the sparse CNN grows approximately linearly during inference with respect to the image resolutions. During training, the memory usage remains clearly below increases in image resolution - an x8 increase in voxel number leads to less than x8 increase in memory requirements. Our study demonstrates the effectiveness of using a sparse CNN for skull reconstruction tasks, and our findings can be applied to other spatially sparse problems. We proof this by additional experimental results on other sparse medical datasets, like the aorta and the heart. Project page at https://github.com/Jianningli/SparseCNN

Hosted file

sparse cnn.mp4 available at https://authorea.com/users/683775/articles/678613-sparse-convolutional-neural-networks-for-medical-image-analysis

Sparse Convolutional Neural Networks for Medical Image Analysis

Jianning Li, Christina Gsaxner, Antonio Pepe, Dieter Schmalstieg, Jens Kleesiek, and Jan Egger

***Abstract*—Traditional convolutional neural network (CNN) methods rely on dense tensors, which makes them suboptimal for spatially sparse data. In this paper, we propose a CNN model based on sparse tensors for efficient processing of large and sparse medical images. In contrast to a dense CNN that takes the entire voxel grid as input, a sparse CNN processes only on the non-empty voxels, thus reducing the memory and computation overhead caused by the sparse input data. We evaluate our method on two clinically relevant skull reconstruction tasks: (1) given a defective skull, reconstruct the complete skull (i.e., skull shape completion), and (2) given a coarse skull, reconstruct a high-resolution skull with fine geometric details (shape super-resolution). Our method outperforms the state of the art in the skull reconstruction task quantitatively and qualitatively, while requiring substantially less memory for training and inference. We observed that, on the 3D skull data, the overall memory consumption of the sparse CNN grows approximately linearly during inference with respect to the image resolutions. During training, the memory usage remains clearly below increases in image resolution - an $\times 8$ increase in voxel number leads to less than $\times 4$ increase in memory requirements. Our study demonstrates the effectiveness of using a sparse CNN for skull reconstruction tasks, and our findings can be applied to other spatially sparse problems. We proof this by additional experimental results on other sparse medical datasets, like the aorta and the heart. Project page at https://github.com/Jianningli/SparseCNN.**

***Index Terms*—sparse convolutional neural network, CT, medical image segmentation, deep learning, shape completion, super-resolution, cranioplasty**

I. Introduction

One of the challenges of transferring recent advances in 3D shape analysis to the medical field is that the 3D objects in typical benchmark datasets are of small to moderate sizes. Thus, memory efficiency is often not a primary concern. When applied to medical images, these algorithms often exceed available memory, even on a high-end GPU with many Gigabytes of memory. For example, the 3D models (e.g., chairs, cars, airplanes, etc.) in ShapeNet collection typically consist of a few thousand points, while a typical high-resolution 3D CT scan yields millions of points when converted to a point cloud representation [1].

An obvious opportunity to address the memory issues lies in exploiting spatial sparsity of the 3D data. Some medical data sets, such as the skull, are inherently sparse, with voxel occupancy rates as low as 10%. Since only non-empty voxels carry geometric information of the 3D shape, a sparse convolutional neural network (CNN) [2–5] can save both memory and computational effort.

In our work, we construct a sparse CNN using the *Minkowski Engine* [5], which was originally designed for spatio-temporal tensors of 4D and up. We demonstrate how to apply the same principles to sparse, binary volumetric data. To that aim, we evaluate our sparse CNN on two skull reconstruction tasks: skull shape completion and skull shape super-resolution. With sparse CNN, the skull images can be processed in their original resolution ($512 \times 512 \times Z$, where Z is the number of axial slices in a head CT scan) with moderate memory requirement. Results show the superiority of sparse CNN over conventional dense CNN in terms of both runtime performance and memory requirements on sparse data.

This paper is an extension of our submission [6] to the AutoImplant 2021 challenge [1]. [6] first demonstrated that it is feasible to use sparse CNN in skull reconstruction tasks and empirically analysed its advantages over regular CNN. Compared to [6], the major improvements of this work are summarized as followed:

- Only the edges of the skulls were used in [6], as the available GPU memory is rather low (6GB). In this work, we can use the whole dense skulls thanks to the extended GPU capacity (12GB).
- The superiority claim in [6] is substantiated by experimental evidence in our work
- Besides shape completion on skull images, we show experimentally in this work that sparse CNN is effective in other skull reconstruction tasks such as skull shape super-resolution, and on additional sparse medical images besides the skulls.

II. Related work

a) Shape completion: Shape completion refers to the process of restoring the missing regions of an object in the format of point clouds [7], meshes [8] or voxel grids [9–11]. Using voxel grids for completion takes the advantage of existent and well-established CNN architectures, such as auto-encoders, which are designed to process *images*. However, an object (e.g., in the form of a points cloud) has to be voxelized to a high-resolution voxel grid in order to preserve the geometric details. The use of a voxel grid in deep learning is expensive, as memory requirements grow cubically with respect to resolution. The work of Han et al. [9] and Dai et

This work was supported by the REACT-EU project KITE (Plattform für KI-Translation Essen).

J. Li, J. Kleesiek and J. Egger are with the Institute for Artificial Intelligence in Medicine, Essen University Hospital, Germany. C. Gsaxner, A. Pepe and D. Schmalstieg are with the Institute of computer graphics and vision, Graz University of Technology, Austria. (e-mail: Jianning.Li@uk-essen.de; Jan.Egger@uk-essen.de). Corresponding authors: Jianning Li and Jan Egger

[1] https://autoimplant2021.grand-challenge.org/

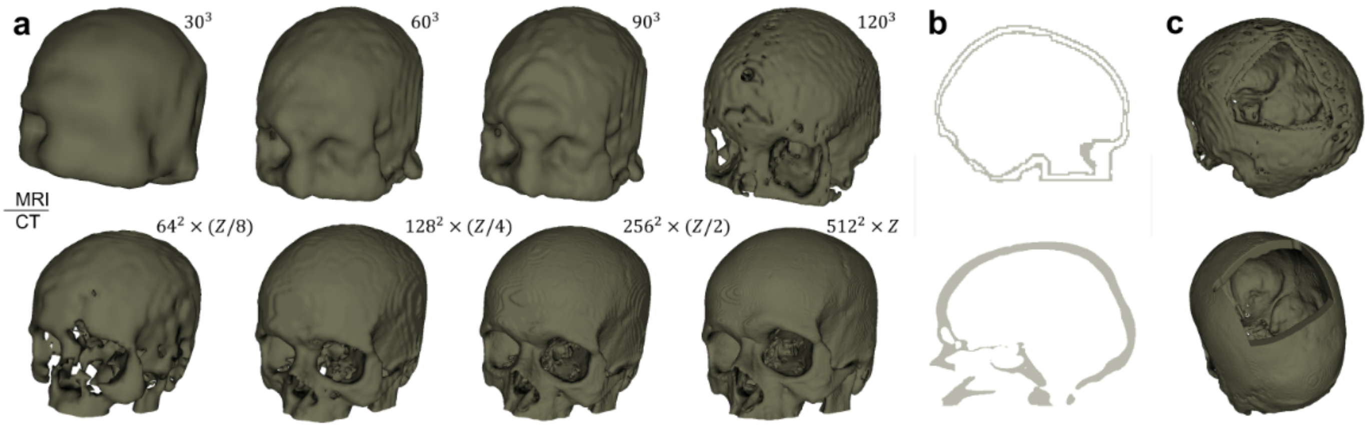

Figure 1: Illustration of the MRI and CT skull dataset (a,b) and the synthetic defects (c) created for skull shape completion.

al. [10] addressed the memory issues by reconstructing high resolution voxel grids in a two-step, coarse-to-fine fashion. Other studies work around the memory issue by using only very coarse voxel grids (e.g., $24 \times 54 \times 24$) for completion [11].

Compared to voxel grids, point clouds are much more lightweight. Yuan et al. [7] proposed a deep learning framework that performs shape completion directly on raw point clouds data without voxelization. However, since point clouds are unstructured and objects of the same size differ in their number of points, deep learning has to deal with irregular memory access [12, 13]. These CNN methods for shape completion generally used the auto-encoder architecture and its variants.

b) Skull shape completion and clinical implications: Skull shape completion has important applications in craniofacial implant design [14, 1, 15]. The skull images are segmented as binary voxel grids from high-resolution CT scans, typically at a resolution of $512 \times 512 \times Z$. In CNN applications, the size of such skull images significantly exceeds the memory capacity of a standard desktop GPU. Previous methods either downsample [16] or resample [17, 18] the skull images to a smaller, intermediate size, or use a patch-wise training and inference strategy [1]. Li et al. [19] proposed a two-step, coarse-to-fine framework that generates high-resolution implants with reduced memory usage. All these methods are far from optimal, as downsampling or resampling inevitably results in image quality degradation and, consequently, deformation of the skull shape. The two-step method proposed by Li et al. [19] is not end-to-end trainable. The patch-based approach requires a tailored training strategy to make sure that the CNN captures the overall shape distribution of the human skull [1]. Besides, it was reported in [1] that the reconstructed high-resolution skulls would appear *patchy* due to the incongruency around the borders of the individual patches. Furthermore, [19, 20] also showed experimentally that a network would be more likely to learn the overall shape distributions of the skulls when given an entire skull as input[2], compared to given only a portion (e.g., a bounding box [19] or a patch [20]) of the skull. The full-image context helps increase a network's robustness against defect patterns and generalizability. On this account, an ideal CNN for skull reconstruction should take the entire high-resolution skull images as input and output the reconstructed skulls or implants in their original resolutions.

c) Data spatial sparsity and sparse CNN: In a recent approach [20], the authors adopted a hash table to exploit the sparse and binary [3] structure of the skull images to reduce the reconstruction time and memory consumption. Instead of the entire skull volume, the method reconstructs only the non-zero voxels and stores them as bit-strings, so that each voxel occupies only one bit of memory. This is a non-CNN approach and requires that voxel coordinates are stored during reconstruction to maintain the spatial relationship among the reconstructed voxels.

In this paper, we propose to take the advantage of such spatial sparsity of the skull data to reduce memory consumption using only sparse convolutions. Note that by "sparse CNN" we mean a CNN architecture made for sparse input data (like the skull) and not a compressed CNN with sparse (e.g., mostly zero) parameters [21–24].

This makes our approach conceptually similar to methods which apply a CNN to 3D shapes at high resolutions, such as Riegler et al. [3] and Wang et al. [2]. Both used an octree representation for 3D shapes and proposed octree-based convolutions. Graham et al. [4, 25] and Choy et al. [5] proposed sparse convolutions defined on the non-empty points in an object. During execution, features are extracted only from these non-empty locations, such that the zero-valued background does not take up memory and computation resources.

III. Data generation

We used two public skull datasets in our study, namely, the MRI skull dataset from the Human Connectome Project (HCP)[4] and the CT skull dataset from the Task 3 of the AutoImplant 2021 challenge.

[2] Due to memory restrictions, [19, 20] used low-resolution images for experiments on the entire skulls.

[3] By sparse, the authors mean that the percentage of the non-zero voxels is rather low in a skull volume. By binary, the voxels in a skull volume have only two status: zero (background voxels) and one (the non-zero voxels that contribute to the skull geometry).

[4] https://humanconnectome.org/study/hcp-young-adult/document/1200-subjects-data-release/

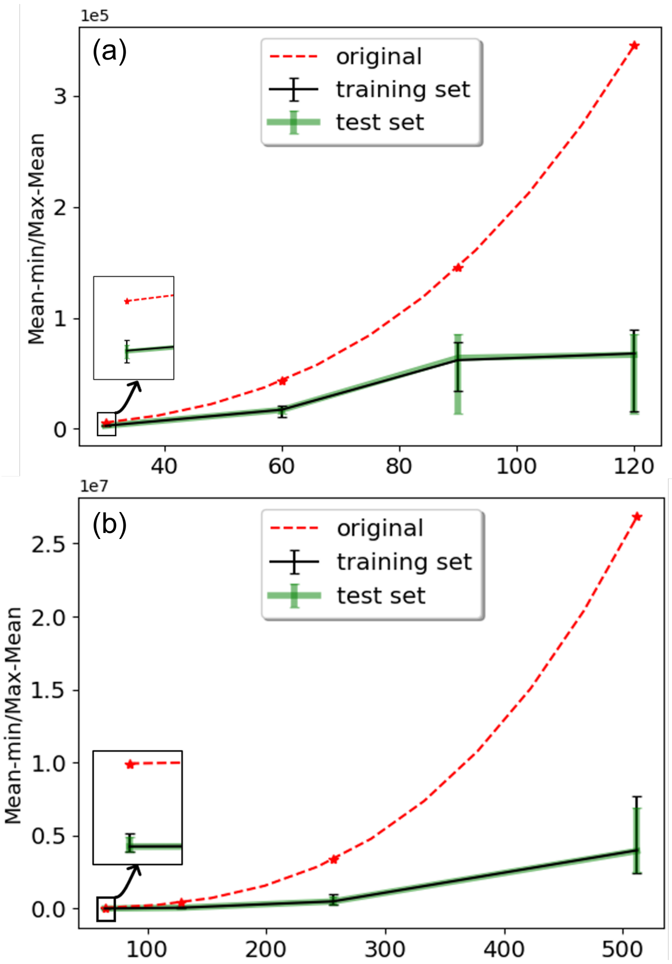

Figure 2: Memory occupancy of the MRI (a) and CT (b) skull datasets at different resolutions. For the *original* skull data, the memory occupancy was down-scaled by a factor of ten and five for the MRI and CT dataset, respectively, for the plots. The X axis shows the resolutions and the Y axis shows the min, max and mean related to the number of voxels.

MRI Dataset: The HCP dataset originally contains 1113 structural MRI scans, and 200 of them were selected in our study (100 for training and 100 for evaluation). The BrainSuite (http://brainsuite.org/) software was used to extract the skull surfaces from the scans [26]. Note that the program extracts only the inner and outer skull surfaces, and therefore the resulting skulls meshes are *hollow* from within (Figure 1 (b)). The skull meshes were further voxelized to binary grid representation at various resolutions: 30^3, 60^3, 90^3 and 120^3 (Figure 1 (a)).

CT Dataset: In CT scans, structures can be distinguished based on gray values, and therefore the bony skull can simply be extracted using thresholding, resulting in binary voxel grids of resolution $512 \times 512 \times Z$ (Z varies for different scans). The resulting skulls are *solid*, in contrast to the hollow MRI skulls. The dataset contains 100 skulls for training and 100 for evaluation (the 10 out-of-distribution test cases are not included here). We also created the multi-resolution representation of the CT skulls at $64 \times 64 \times (Z/8)$, $128 \times 128 \times (Z/4)$ and $256 \times 256 \times (Z/2)$, as illustrated in Figure 1 (a).

For both the datasets, a portion of the skull bone (around the cranium area) was removed to simulate the surgical procedure of craniotomy for the experiments on skull shape completion (Figure 1 (c)). Figure 2 shows a comparison of the memory occupancy between the original skull voxel grids and the non-zero voxels, for the MRI (Figure 2 (a)) and CT dataset (Figure 2 (b)) at various resolutions specified above. Note that the plots use the number of voxels to represent the overall memory occupancy directly, as each voxel occupies a constant space[5]. The plots show that the memory usage of the original skull data grows cubically with respect to image resolutions, while for the valid voxels, memory usage exhibits approximate linear growth in comparison. Intuitively, a sparse CNN relying only on the valid voxels would be more efficient in terms of memory and computation than a dense CNN that takes the entire voxel grids as input.

IV. Methods

We uses the *Minkowski Engine* proposed by Choy et al. [5] as the backbone of a sparse CNN. *Minkowski Engine* is originally designed as a general-purpose tool for the analysis of 4D spatio-temporal data and uses sparse tensors as the basic data type. A sparse tensor $\mathscr{F}$ is a generalized representation of a sparse matrix in which most of the points are empty (zero). A third order sparse tensor can be expressed as:

$$\mathscr{F}(x_i, y_i, z_i) = \begin{cases} f_i, & (x_i, y_i, z_i) \in \mathcal{C} \\ 0, & \quad others \end{cases} \tag{1}$$

where $\mathcal{C}$ is the coordinate matrix (row-wise concatenation of coordinates) of the non-empty points and $f_i \in \mathbb{R}^{N_F}$ is the non-empty value at coordinate (x_i, y_i, z_i). N_F is the number of channels at this point. $\mathcal{F} = \{f_1, f_2, ..., f_i, f_{i+1}...\}$ is the feature vector. Sparse CNN relies only on $\mathcal{C}$ and $\mathcal{F}$ for feature computation. In our study, we use sparse CNN specifically on sparse binary volumes of static data, i.e., the skull images, which are typical examples of sparse tensors, since the majority of voxels in a skull image are zero. The input of the sparse CNN consists of a coordinate matrix $\mathcal{C}_{in}$ and the associated feature vectors $\mathcal{F}_{in}$:

$$\mathcal{C}_{in} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ ... & ... & ... \\ x_N & y_N & z_N \end{bmatrix}, \mathcal{F}_{in} = \begin{bmatrix} 1 \\ 1 \\ ... \\ 1 \end{bmatrix} \tag{2}$$

Here, N is the number of non-zero voxels in a skull image. Note that the coordinates we used in our study refer to voxel grid coordinates (i.e., from [0, 0, 0] to [512, 512, Z]) instead of the world coordinates of point clouds. Since the skull data are binary, and the number of channels per voxel is one ($N_F = 1$), the feature vector has a format of $\mathcal{F}_{in} \in \mathbb{R}^{N \times 1}$, and the elements in $\mathcal{F}_{in}$ are all 1. For three-dimensional voxel grid coordinates, $\mathcal{C}_{in} \in \mathbb{Z}^{N \times 3}$. A general data pre-processing step for using the sparse CNN is to format the input and ground truth skull images according to Equation 2.

[5] The MRI dataset was stored as *int8* and the CT dataset was stored as *int32*.

Similar to existing CNN methods for shape completion, we use an auto-encoder architecture for the task, but we replaced the conventional dense convolutional layers with sparse convolutional layers [5]. Table I shows the configuration of each layer in the sparse CNN used for experiments. ch is a list of the channel numbers for each layer. As the number of output channels (C^{out}) in each layer is no longer constant 1 as in the input skull image, we use $\mathcal{F}_{in}^{i} \in \mathbb{R}^{C_{i-1}^{out}}$ as a general notation for the output (i.e., the feature vector) of the intermediate layer i. The convolution operation at a coordinate $D \in \mathbb{Z}^3$ in the sparse CNN can therefore be defined similar to that of the traditional dense CNN:

$$\mathcal{F}_{in}^{i+1}(D') = \sum w^i \mathcal{F}_{in}^{i}(D) + b_i \tag{3}$$

where $w^i \in \mathbb{R}^{C_i^{out} \times C_{i-1}^{out}}$ and b_i is the weight matrix and bias of intermediate layer i. $D' \in \mathbb{Z}^3$ is the corresponding coordinate in layer $i+1$ mapped from D. Note that, unlike a traditional dense CNN that operates on regular voxel grids sequentially, a sparse CNN requires specifying a coordinate mapping in order to know how D is mapped to D', as the non-zero voxels can be distributed arbitrarily, and, by extracting only the non-zero voxels, the spatial context within an image is lost. For such coordinate mapping, *Minkowski Engine* uses a pair of voxel indices from the input and ground truth images to memorize the mapping relationship as in a regular voxel grid, leading to coordinates-related computation overhead, comparable to Li et al. [20]. In *Minkowski Engine*, the coordinates and the voxel indices were stored in a hash table (the hash function used is *FNV64-1A*), where the coordinates were used as hash keys to retrieve the original voxel indices of the associated elements in a feature vector. Even if the hash table is not directly involved in feature computation, they determine how an element from the input feature vector is mapped to an element computed according to Equation 3 in the output feature vector.

A. Shape completion

For skull shape completion, the input is a defective skull and the output (ground truth) is the complete skull. It can be divided into two sub-tasks: reconstructing the original defective skull $\{\mathcal{C}_{in}, \mathcal{F}_{in}\}$ and restoring the missing skull bone (i.e., the implant) $\{\mathcal{C}_{imp}, \mathcal{F}_{imp}\}$:

$$\mathcal{C}_{out}^{sc} = \begin{bmatrix} \mathcal{C}_{in} \\ \mathcal{C}_{imp} \end{bmatrix}, \mathcal{F}_{out}^{sc} = \begin{bmatrix} \mathcal{F}_{in} \\ \mathcal{F}_{imp} \end{bmatrix} \tag{4}$$

where

$$\mathcal{C}_{imp} = \begin{bmatrix} x_{N+1} & y_{N+1} & z_{N+1} \\ x_{N+2} & y_{N+2} & z_{N+2} \\ \cdots & \cdots & \cdots \\ x_{N+M} & y_{N+M} & z_{N+M} \end{bmatrix}, \mathcal{F}_{imp} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \tag{5}$$

M is the number of non-zero voxels in the generated set of coordinates $\mathcal{C}_{imp}$. $\mathcal{F}_{imp} \in \mathbb{R}^{M \times 1}$, and the elements in $\mathcal{F}_{imp}$ are all 1. According to Equation 4, the sparse CNN needs to generate new sets of coordinates $\mathcal{C}_{imp}$ at which the values are non-zero, for the skull shape completion task ($M \neq 0$ such that $\mathcal{C}_{in} \subset \mathcal{C}_{out}^{sc}$). The generative sparse tensor decoder in Table I are composed of generative transposed convolutional layers [27] that are capable of generating new non-zero points absent in the input. Given a sparse tensor $\mathscr{F}$ as input, the output of a transposed convolution $\mathscr{F}'$ can be written as:

Encoder			Decoder		
C^{in}	C^{out}	Ks	C^{in}	C^{out}	Ks
1	ch[0]	3	***ch[6]**	**ch[5]**	**4**
*ch[0]	ch[1]	2	ch[5]	ch[5]	3
ch[1]	ch[1]	3	ch[5]	1	1
*ch[1]	ch[2]	2	***ch[5]**	**ch[4]**	**2**
ch[2]	ch[2]	3	ch[4]	ch[4]	3
*ch[2]	ch[3]	2	ch[4]	1	1
ch[3]	ch[3]	3	***ch[4]**	**ch[3]**	**2**
*ch[3]	ch[4]	2	ch[3]	ch[3]	3
ch[4]	ch[4]	3	ch[3]	1	1
*ch[4]	ch[5]	2	***ch[3]**	**ch[2]**	**2**
ch[5]	ch[5]	3	ch[2]	ch[2]	3
*ch[5]	ch[6]	2	ch[2]	1	1
ch[6]	ch[6]	3	***ch[2]**	**ch[1]**	**2**
-	-	-	ch[1]	ch[1]	3
-	-	-	ch[1]	1	1
-	-	-	***ch[1]**	**ch[0]**	**2**
-	-	-	ch[0]	ch[0]	3
-	-	-	ch[0]	1	1
-	-	-	ch[0]	1	1
-	-	-		sigmoid	

Table I: Configuration (number of input channels C^{in}, output channels C^{out} and kernel size Ks) of each layer in the encoder and decoder of the sparse CNN. The generative transposed convolutional layers are marked bold. Layers with stride 2 are marked with *.

$$\mathscr{F}'[x,y,z] = \sum_{i,j,k \in \mathcal{N}(x,y,z)} \mathcal{W}[x-i, y-j, z-k] \mathscr{F}[i,j,k] \tag{6}$$

where $(x,y,z) \in \mathcal{C}'$ and $(i,j,k) \in \mathcal{C}$. $\mathcal{W}$ is the kernel weight. $\mathcal{C}$ and $\mathcal{C}'$ are the input and output coordinate matrix respectively, and they have the following relationship:

$$\mathcal{C}' = \mathcal{C} \otimes [-K, ..., K]^3 \tag{7}$$

$\otimes$ denotes outer-product. $2K+1$ is the transposed convolution kernel size. A point generated by a transposed convolution (x,y,z) has the following constraint with the input coordinate i, j and k according to Equation 6:

$$\begin{aligned} &\mathcal{N}(x,y,z) = \\ &\left\{(i,j,k) \middle| |x-j| \leqslant K, |y-j| \leqslant K, |z-j| \leqslant K\right\} \end{aligned} \tag{8}$$

We can see from Equation 6 - Equation 8 that using a kernel size greater than two would expand the span (e.g., $[-K, K]$) of the input coordinates, allowing a transposed convolution to dynamically generate new non-zero points for generative tasks like shape completion. In our specific task, the generative sparse tensor decoder in Table I is trained to generate M new points, while maintaining the original input coordinates.

Each transposed convolution layer in Table I is followed by a *pruning* layer that prunes out undesirable new points, which is essential for maintaining a low memory and computation cost during the generative process. During training, the ground truth masks teach the network when to keep or prune a point. During inference, the ground truth masks are unavailable. The network prunes a point if its feature value is lower than a pre-defined threshold τ. In our network, we choose $\tau = 0$.

B. Shape super-resolution

Skull shape super-resolution refers to the process of transforming a (completed) coarse binary skull shape to its smooth high-resolution representation with fine geometric details. The input is the completed skull at a low resolution $\mathbb{Z}^a$, and the output is the same completed skull at a higher resolution $\mathbb{Z}^b$, $a < b$.

Note that for the skull super-resolution task, the coarse and high-resolution skull (i.e., the ground truth) have to be in the same coordinate system for the coordinate mapping in the sparse CNN to work properly, meaning that the coarse skull image needs to be up-scaled to the same size as the target high-resolution image, i.e., $a = b$.[6] By $a = b$ we do not mean that the input and the ground truth have the same *resolution* from the perspective of image quality. Rather, we mean that the input is interpolated to the same size as the ground truth. The up-scaled input still appears blurry and coarse, and lacks geometric details.

According to [20], the difference between a (up-scaled) coarse skull voxel grid and a high-resolution voxel grid is simply the arrangement patterns of the zero and non-zero voxels, and, by rearranging the voxels, a coarse skull shape can be upgraded to the high-resolution representation. The total number of non-zero voxels between the two types of skulls shows no statistical differences. Therefore, we use the following to represent the ground truth coordinate matrix $\mathcal{C}_{out}^{sr}$ and feature vector $\mathcal{F}_{out}^{sr}$ for the super-resolution task:

$$\mathcal{C}_{out}^{sr} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \cdots & \cdots & \cdots \\ x_{N_0} & y_{N_0} & z_{N_0} \\ \cdots & \cdots & \cdots \\ x'_N & y'_N & z'_N \end{bmatrix}, \mathcal{F}_{out}^{sr} = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \\ \cdots \\ 1 \end{bmatrix}. \tag{9}$$

If we assume that the input $\mathcal{C}_{in}$ and the ground truth share N_0 common points ($N_0 < N$), $N - N_0$ non-zero points in the input need to be pruned while $N - N_0$ new non-zero points need to be generated. Therefore, the sparse CNN specified in Table I is still applicable to the super-resolution task.

C. Memory usage and computation complexity

The memory consumption of a neural network comes primarily from the following sources during training time: (1) input and ground truth image batches, (2) the output of the intermediate layers (forward pass), (3) network parameters, (4) memory usage from back-propagation (errors and gradients at each parameter) and (5) optimizers. In test time, the parameters of the network, input image batches and intermediate layers' output are the main sources of memory usage. In our study, we compare the memory consumption of sparse and dense CNN when the networks have the same configurations (Table I) and number of parameters N_{param}. For both dense and sparse CNN configurations, N_{param} can be estimated as

$$N_{param} = \sum_i \mathcal{C}_i^{out} \times \mathcal{C}_i^{in} \times Ks^3 + \mathcal{C}_i^{out}. \tag{10}$$

The number of bias in layer i is the same as the number of output channels of the layer $\mathcal{C}_i^{out}$. Assuming that the parameters are stored as *float32* (32-bit), sparse and dense CNN consume the same amount of memory in storing these parameters. However, the input and ground truth for a dense CNN are the original voxel grids, while, for a sparse CNN, only the valid non-zero voxels are required, and thus a sparse CNN consumes significantly less memory than dense CNN in loading the input and ground truth image batches, as shown in Figure 2. Similarly, the size of the output N_{f^i} corresponding to intermediate layers i is linear to the feature dimension of the $(i-1)th$ layer $N_{f^{i-1}}$ and is calculated as

$$N_{f^i} = \frac{1}{s}(N_{f^{i-1}} + 2p - Ks), \tag{11}$$

where p and s are the padding and stride size, respectively. According to Equation 11, the memory consumption of the intermediate layers' output is also linear to the input image size (Figure 2).

The memory consumption related to back-propagation and optimizer is tricky to calculate. In our study, we estimate the overall GPU memory usage during training using the *nvidia-smi* command provided by NVIDIA. We query the system GPU memory usage at 50-millisecond intervals for N_{train} training iterations (N_{train} is the number of training samples, and the batch size was set to 1) and take the average of all the queried values as the final amount of memory consumed for training, considering that the number of non-zero voxels are different for each training sample. The static memory occupancy that is not caused by training the network was subtracted from the measurement. For inference, we used the same method except that the measurement was taken when the network loaded the trained parameters and was run on the test set.

Floating points operations (FLOPS) is commonly used to measure computational complexity of a CNN. The FLOPS consumed in CNN layer i is the product of N_{f^i}, Ks and $\mathcal{C}_i^{out} \times \mathcal{C}_i^{in}$. Given the same network configurations ($\mathcal{C}_i^{out}$, $\mathcal{C}_i^{in}$, Ks), the FLOPS are linear to N_{f^i} and thus to the input image size (Figure 2). The sparse CNN therefore is significantly faster than a dense CNN in both training and inference time under the same configurations.

V. Experiments and Results

We trained the sparse CNN (Table I) for two tasks: The first task is skull shape completion on the CT and MRI skull dataset at different resolutions (30^3, 60^3, 90^3 and 120^3 for the MRI dataset and $64^2 \times (Z/8)$, $128^2 \times (Z/4)$, $256^2 \times (Z/2)$ and $512^2 \times Z$ for the CT dataset). For the CT dataset, ch is set to $ch1 = [8, 8, 16, 16, 32, 32, 64]$ (0.435M parameters), for the MRI dataset, ch is set to $ch2 = [22,$

[6] The network would fail to converge when, for example, the input is of resolution $64 \times 64 \times (Z/8)$, while the ground truth is of resolution $256 \times 256 \times (Z/2)$.

Table II: Quantitative results - DSC (top row) and RE (%, second row) for the skull shape completion task.

MRI				CT (sparse)						CT (dense)		
30	60	90	120	64($ch1$)	64($ch2$)	128 ($ch1$)	128 ($ch2$)	256	512	64	128	256
0.8794 2.5593	0.9915 0.1324	0.9920 0.1333	0.9879 0.1095	0.9798 0.2237	0.9859 0.1561	0.9876 0.1423	0.9892 0.1229	0.9876 0.1443	0.9903 0.1144	0.4801 7.9719	0.4928 6.0845	0.6069 4.8014

Table III: Quantitative results - DSC (first row) and RE (second row, %) for the skull shape super-resolution task.

$64 \rightarrow 128$	$64 \Rightarrow 128$	$64 \rightarrow 256$	$64 \Rightarrow 256$	$64 \rightarrow 512$	$64 \Rightarrow 512$	$128 \rightarrow 256$	$128 \Rightarrow 256$	completion
0.8750 1.3821	0.8359 1.8685	0.8779 1.3589	0.8359 1.8942	0.6640 3.7850	0.6402 4.2358	0.9372 0.7187	0.9146 0.9867	0.9876 0.1443

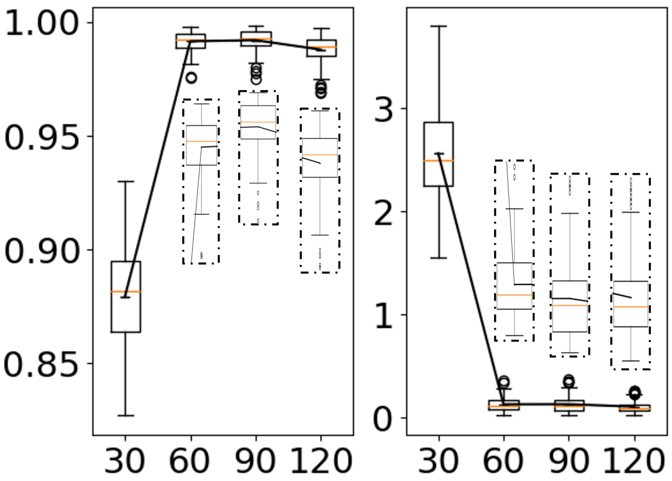

Figure 3: DSC (left) and RE (right, %) on the MRI dataset at different resolutions (30^3, 60^3, 90^3, 120^3) for the shape completion task. The dash-lined boxes contain the zoomed-in boxplots.

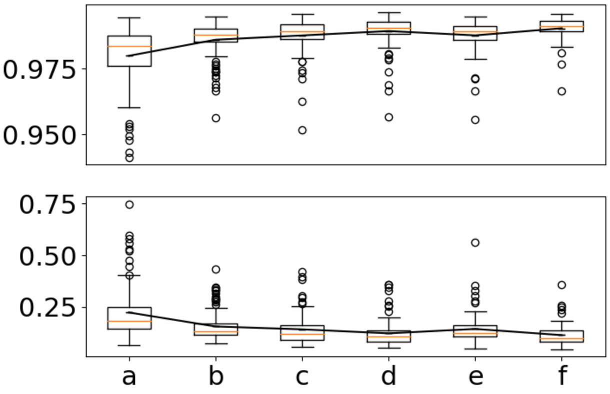

Figure 4: DSC (top) and RE (%, bottom) for the sparse CNN on the CT dataset at different resolutions. (a) 64 ($ch1$) (b) 64 ($ch2$) (c) 128 ($ch1$) (d) 128 ($ch2$) (e) 256 (f) 512.

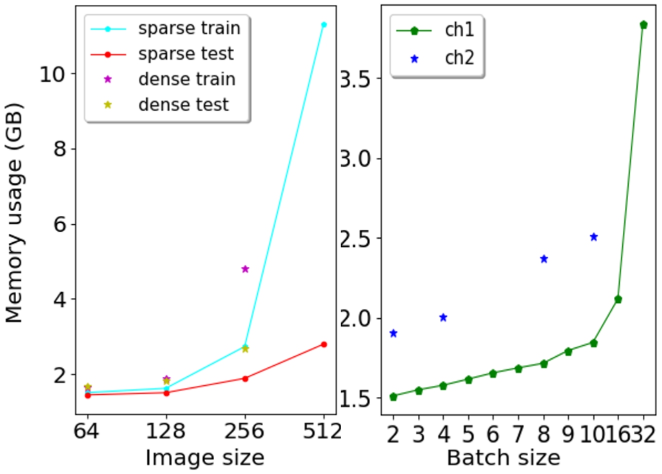

Figure 5: Memory consumption during training and inference for the sparse and dense CNN at different resolutions (left). Memory consumption of sparse CNN with different batch sizes at resolution 64 (right).

Table IV: Comparison of estimated memory consumption (in *GB*) during training and inference between the sparse and dense CNN at different image resolutions.

cat. $\setminus$ I_s	64	128	256	512
sparse train	1.5119	1.6256	2.7341	11.3049
sparse test	1.4519	1.5097	1.8905	2.7993
dense train	1.6543	1.9043	4.8145	-
dense test	1.6699	1.8184	2.6934	-

32, 32, 128, 156, 256, 388] (about 18.14M parameters). The second task is skull shape super-resolution on the CT skull dataset on different scales: $64^2 \times (Z/8) \rightarrow 128^2 \times (Z/4)$, $64^2 \times (Z/8) \rightarrow 256^2 \times (Z/2)$, $64^2 \times (Z/8) \rightarrow 512^2 \times Z$ and $128^2 \times (Z/4) \rightarrow 256^2 \times (Z/2)$. For comparison, we also trained a standard dense CNN with the same configuration as the sparse CNN for the shape completion task on the CT dataset. For both tasks, we used dice similarity coefficient (DSC) and reconstruction error (RE), i.e., the percentage of misclassified voxels, to evaluate the predictions. The sparse CNN was trained using a binary cross-entropy loss $\mathcal{L}_{bce}$:

$$\mathcal{L}_{bce} = y' \cdot log\sigma(y) + (1 - y') \cdot log(1 - \sigma(y)) \tag{12}$$

and the dense CNN was trained using a dice loss $\mathcal{L}_{dice}$ for

Table V: Sparse CNN memory consumption (in *GB*) with different batch sizes at resolution $64^2 \times (Z/8)$ during training.

ch \batch	2	3	4	5	6	7	8	9	10	16	32
$ch1$	1.5119	1.5494	1.5780	1.6164	1.6557	1.6867	1.7151	1.7950	1.8459	2.1180	3.8395
$ch2$	1.9071	-	2.0054	-	-	-	2.3729	2.3232	2.5116	-	-

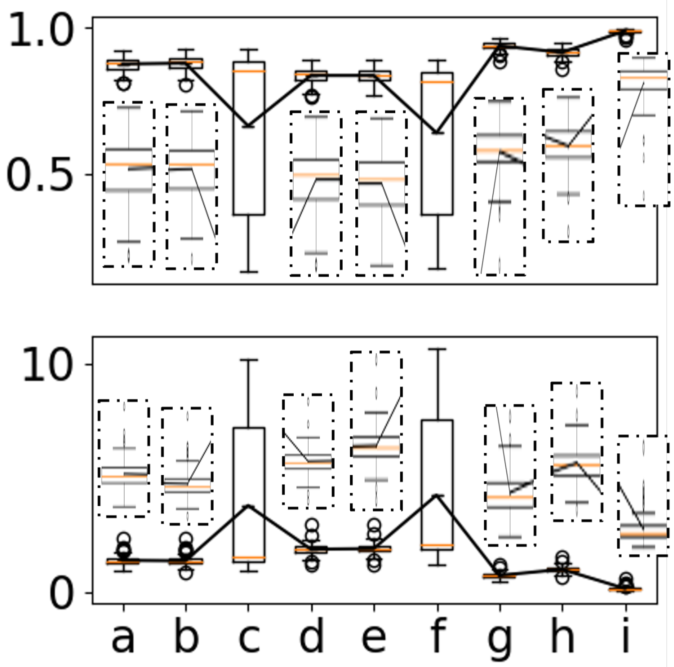

Figure 6: DSC (top) and RE (bottom, %) for the super-resolution task on the CT dataset. (a) 64 → 128 (b) 64 → 256) (c) 64 → 512 (d) 64 ⇒ 128 (e) 64 ⇒ 256 (f) 64 ⇒ 512 (g) 128 → 256) (h) 128 ⇒ 256 (i) shape completion at 256. The dash-lined boxes contain the zoomed-in boxplots.

the background ($i = 0$) and the target ($i = 1$):

$$\mathcal{L}_{dice} = -2 \sum_{i=0}^{1} \frac{\sum y^i \circ y'^i}{\sum y^i \circ y^i + \sum y'^i \circ y'^i}, \tag{13}$$

σ is a sigmoid non-linearity. y and y' denote the predictions and the ground truth, respectively. $\circ$ denotes element-wise multiplication between two matrices.

Table II shows the quantitative evaluation results (mean DSC and RE) for the shape completion task. In Table II, we also reported a performance comparison of the sparse CNN with different numbers of parameters at resolutions $64^2 \times (Z/8)$ and $128^2 \times (Z/4)$. Results indicate that increasing the model complexity of the sparse CNN would also lead to increased prediction accuracy, a phenomenon well observed in traditional dense CNN models. It is worth noting that, using a sparse CNN, we are able to train on the CT skull images at their full resolutions ($512^2 \times Z$) and the results are promising with over 0.99 DSC and less than 0.12% reconstruction error (e.g., in a $512^2 \times 256$ image, only 76772 voxels are misclassified on average). In contrast, GPU memory restrictions made training on the $512^2 \times Z$ image resolution using a dense CNN unsuccessful. Furthermore, the quantitative results of the dense CNN were significantly worse than the sparse CNN, as can be seen in Table II. Note that the shape completion results in Table II are not directly comparable to the AutoImplant challenge results for two reasons: 1) for a fair comparison, the dense CNN used the same vanilla network configuration as the sparse CNN, while the challenge submissions used more complex (and different) dense network architectures combined with tailored pre- and post-processing (e.g., data augmentation) to achieve the results [14]. Besides, Table II reported the results at resolutions $64^2 \times (Z/8)$, $128^2 \times (Z/4)$ and $256^2 \times (Z/2)$ for the dense CNN, while the challenge reported the results at resolution $512^2 \times Z$. 2) the results reported in Table II apply to the skulls while the challenge results apply to the implants [14]. Therefore, Table II is solely to show a comparison of sparse and dense CNN under one vanilla setting. To provide an external comparison for the proposed sparse CNN, we refer to [20], in which a dense network with over 82M parameters was trained on the same CT dataset for skull shape completion. DSC from three variants of the method was reported: 0.7547 for interpolation, 0.7529 for voxel rearrangement and 0.8587 for patch-based training and inference. Quantitatively, the sparse CNN with only 0.435M parameters performs significantly better (DSC of 0.9903). Figure 3 and Figure 4 shows the DSC and RE distributions over the test sets on the MRI and CT skull datasets, respectively.

Figure 5 shows a comparison of the estimated memory consumption of dense and sparse CNN at different image resolutions during training and inference, as well as a comparison of memory consumption of sparse CNN with different batch sizes at image resolution $64^2 \times (Z/8)$ during training. Table IV and Table V report the estimated memory usage (in *GB*). With each increase in the image resolution, the image size increases cubically (×8). During training, the memory consumption of the sparse CNN increases in an approximately linear manner when the image resolution is no more than $256^2 \times (Z/2)$. At $512^2 \times Z$ resolution, the memory usage quadruples (×4). For the dense CNN, the memory usage demonstrates non-linear growth. During inference, the memory usage of the sparse CNN increases linearly at all resolutions, and the memory consumption of the sparse CNN increases linearly with respect to batch size. Furthermore, the sparse CNN with $ch2$ channels possesses over 40 times the parameters than with $ch1$ channels, whereas the memory increases by less than two times. We take this as indication that, for a sparse CNN, raising the model complexity to improve the prediction accuracy does not cause dramatic increases in memory usage. Figure 7 and Figure 8 show the qualitative completion results on the MRI and CT datasets at different resolutions.

We use the average GPU execution time per skull image to portray the runtime speed of the sparse CNN. For training, we measure the duration of training for 100 iterations and compute the average time per iteration. For inference, we measure the time it takes to run on the entire test set (100 images). Batch size is set to one in both cases. We experimented on the CT data using the shape completion model ($ch1$). At resolution $64^2 \times (Z/8)$, $128^2 \times (Z/4)$ and $256^2 \times (Z/2)$, the training/inference time (s) per image is roughly $0.28/0.22$, $0.32/0.30$ and $0.71/0.54$, excluding data loading. We can see that both training and inference time increases linearly with respect to resolutions. Note that the time measured the same way for the dense CNN is not directly comparable to that of the sparse CNN, as the variable i.e., the amount of computational resources they occupy in runtime, can not be controlled during measurement.

Keep in mind that the time and memory growth reported above is not strictly linear, especially during training at high resolutions. The memory and time overhead includes space and computation reserved for voxel coordinates, coordinate mapping and other implementation-related costs.

Table III shows the quantitative evaluation results for the super-resolution task. In Table III, → represents super-resolution using the sparse CNN, and ⇒ represents up-scaling using interpolation. Figure 6 shows the DSC and RE distributions. We can see that super-resolution with a sparse CNN outperforms interpolation-based up-scaling. Besides, super-resolution directly from the lowest-resolution to the highest resolution (i.e., $64^2 \times (Z/8) \rightarrow 512^2 \times Z$) has the worst results, and the sparse CNN shows better performance at smaller resolution gaps ($128^2 \times (Z/4) \rightarrow 256^2 \times (Z/2)$ is better than $64^2 \times (Z/8) \rightarrow 256^2 \times (Z/2)$). Table III compares super-resolution and shape completion at resolution $256^2 \times (Z/2)$. The results suggest that sparse CNN might be better at the completion task as well.

The qualitative results in Figure 9 further demonstrate the advantages of super-resolution using a sparse CNN. We can see that the missing geometric details in and around the craniofacial area of the coarse skulls can be effectively recovered in the final super-resolution output.

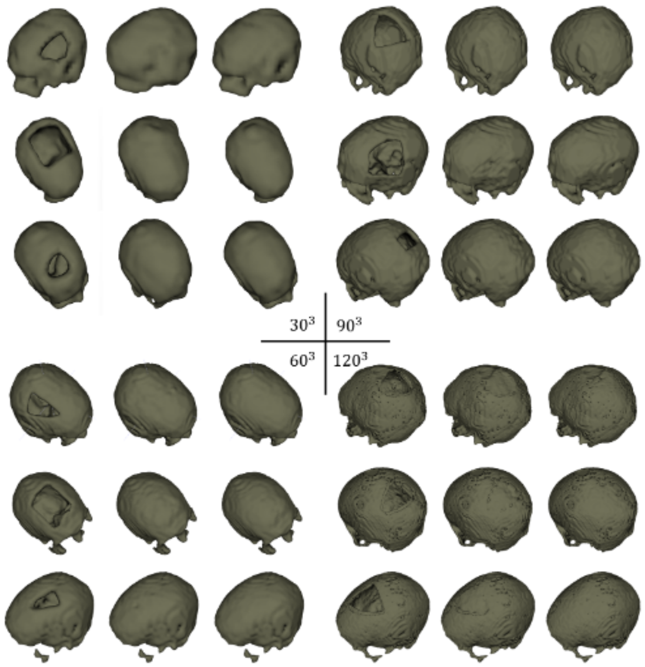

Figure 7: Examples of skull shape completion results with sparse CNN on the MRI skull dataset at different resolutions. The first to third column in each example shows the input defective skull grids, the predictions and the ground truth, respectively.

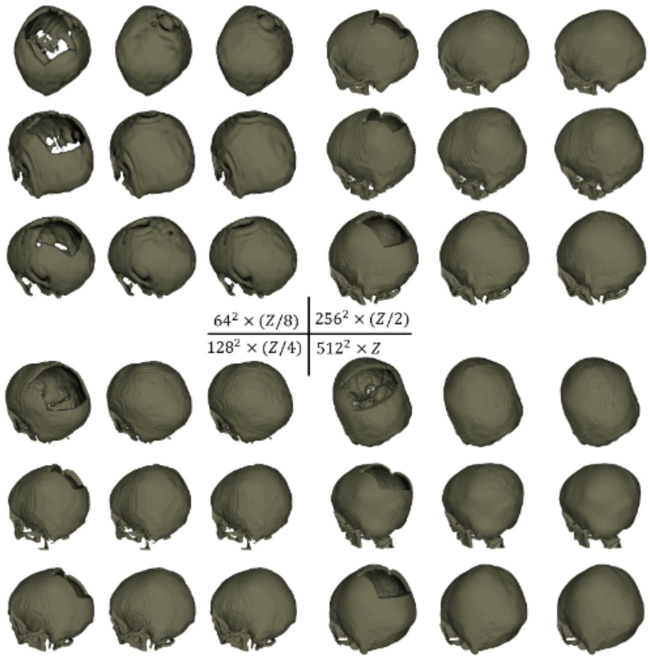

Figure 8: Shape completion results with sparse CNN on the CT skull dataset at different resolutions. The first to third column in each example shows the input defective skull grids, the predictions and the ground truth, respectively.

VI. Discussion, Conclusions and Future Work

In this paper, we have presented a comprehensive evaluation of sparse CNN architectures in two medical image analysis tasks: skull shape completion and skull shape super-resolution. Results show that a sparse CNN significantly outperforms a traditional dense CNN with respect to speed, quality and memory efficiency on sparse data. One of the limitations of current sparse CNN frameworks, such as the *Minkowski Engine* used in our study, is that the voxel coordinates as well as the associated coordinate management tools need to be created and stored to memorize the spatial relationship of the non-zero voxels during convolutions, causing computation overhead in comparison to dense CNN. Another limitation is that, if not initialized properly, the generative transposed convolutional layers might generate a large amount of points and cause false out-of-memory errors during training. Therefore, one future direction worth investigating is to regularize the generative layers or to use shape priors on the output to prevent the network from generating random amount of

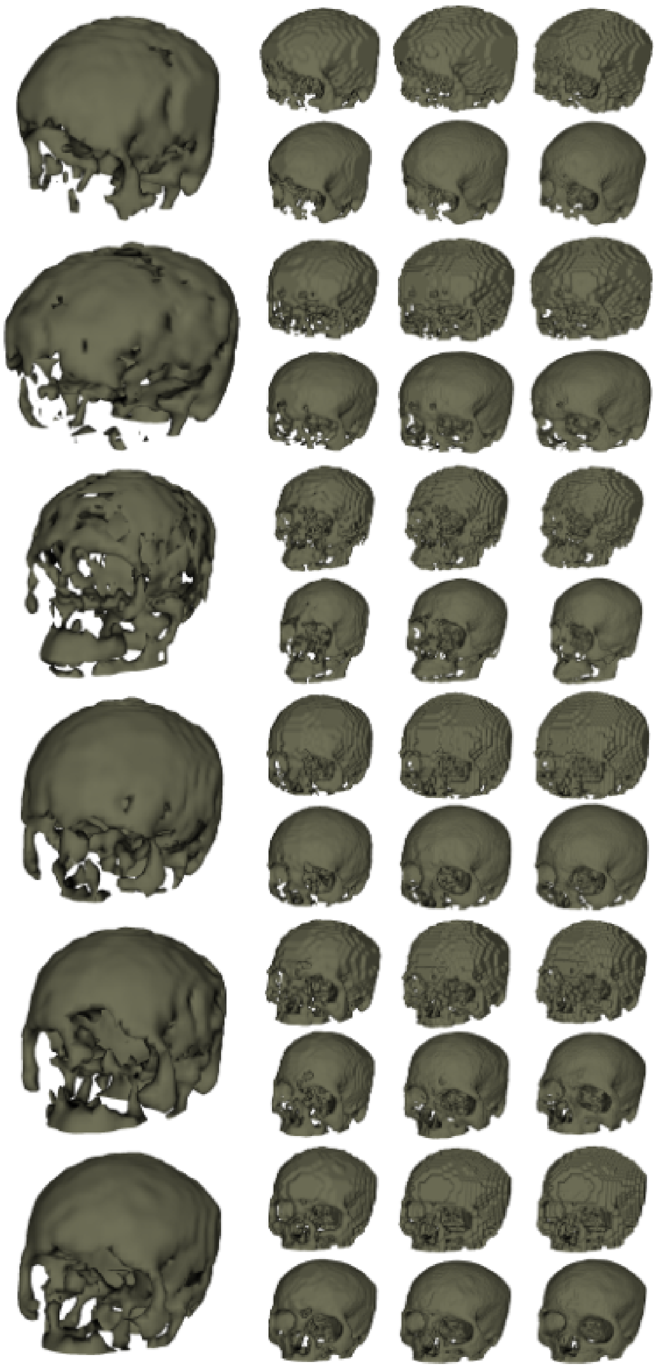

Figure 9: Qualitative results of skull shape super-resolution. The first column shows coarse, completed skulls at 64 resolution. The second to last column show the super-resolution results at 128, 256, 512 resolutions. The first row in each example shows the input of the super-resolution network (upscaled coarse skulls at 128, 256, 512 resolutions), while the second row shows the corresponding network output.

points. Additionally, the sparse CNN failed on the out-of-distribution test set of the AutoImplant Challenge, meaning that the network was overfitting to skull defect patterns and lacking generalizability, even if given a full-image context during training. We presume that the network would fail on real craniotomy skulls as well, since craniotomy defects tend to be more irregular than the synthetic defects used in our shape completion experiments. We have yet to decide on the cause of the failure, and future efforts on this issue are still required. In the supplementary material, we provided additional experiments and results on other spatially sparse medical images, such as the heart, aortic vessels, trachea and esophagus, in a segmentation task. The results indicate that, with moderate increase of computation and memory, the quality of the initial segmentation masks from a dense CNN can be substantially improved using the proposed sparse CNN model.

References

[1] J. Li, G. von Campe, A. Pepe, C. Gsaxner, E. Wang, X. Chen, U. Zefferer, M. Tödtling, M. Krall, H. Deutschmann *et al.*, "Automatic skull defect restoration and cranial implant generation for cranioplasty," *Medical Image Analysis*, p. 102171, 2021. 1, 2

[2] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017. 1, 2

[3] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3577–3586. 2

[4] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232. 2

[5] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084. 1, 2, 3, 4

[6] A. Kroviakov, J. Li, and J. Egger, "Sparse convolutional neural network for skull reconstruction," in *Towards the Automatization of Cranial Implant Design in Cranioplasty II*. Springer, 2021, pp. 80–94. 1

[7] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 728–737. 1, 2

[8] V. Kraevoy and A. Sheffer, "Template-based mesh completion." in *Symposium on Geometry Processing*, vol. 385. Citeseer, 2005, pp. 13–22. 1

[9] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, "High-resolution shape completion using deep neural networks for global structure and local geometry inference," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 85–93. 1

[10] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proceedings of the IEEE Conference on*

Computer Vision and Pattern Recognition, 2017, pp. 5868–5877. 2

[11] D. Stutz and A. Geiger, "Learning 3d shape completion from laser scan data with weak supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1955–1964. 1, 2

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 2

[13] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel cnn for efficient 3d deep learning," *Advances in Neural Information Processing Systems*, vol. 32, pp. 965–975, 2019. 2

[14] J. Li, P. Pimentel, A. Szengel, M. Ehlke, H. Lamecker, S. Zachow, L. Estacio, C. Doenitz, H. Ramm, H. Shi *et al.*, "Autoimplant 2020-first miccai challenge on automatic cranial implant design," *IEEE Transactions on Medical Imaging*, 2021. 2, 7

[15] O. Kodym, M. Španěl, and A. Herout, "Deep learning for cranioplasty in clinical practice: Going from synthetic to real patient data," *Computers in Biology and Medicine*, vol. 137, p. 104766, 2021. 2

[16] J. G. Mainprize, Z. Fishman, and M. R. Hardisty, "Shape completion by u-net: An approach to the autoimplant miccai cranial implant design challenge," in *Cranial Implant Design Challenge*. Springer, 2020, pp. 65–76. 2

[17] F. Matzkin, V. Newcombe, B. Glocker, and E. Ferrante, "Cranial implant design via virtual craniectomy with shape priors," in *Cranial Implant Design Challenge*. Springer, 2020, pp. 37–46. 2

[18] D. G. Ellis and M. R. Aizenberg, "Deep learning using augmentation via registration: 1st place solution to the autoimplant 2020 challenge," in *Cranial Implant Design Challenge*. Springer, 2020, pp. 47–55. 2

[19] J. Li, A. Pepe, C. Gsaxner, G. von Campe, and J. Egger, "A baseline approach for autoimplant: the miccai 2020 cranial implant design challenge," in *Multimodal Learning for Clinical Decision Support and Clinical Image-Based Procedures*. Springer, 2020, pp. 75–84. 2

[20] J. Li, A. Pepe, C. Gsaxner, Y. Jin, and J. Egger, "Learning to rearrange voxels in binary segmentation masks for smooth manifold triangulation," *arXiv preprint arXiv:2108.05269*, 2021. 2, 4, 5, 7

[21] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 806–814. 2

[22] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 27–40, 2017.

[23] Y. Lu, G. Lu, B. Zhang, Y. Xu, and J. Li, "Super sparse convolutional neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4440–4447.

[24] G. Xie, J. Wang, T. Zhang, J. Lai, R. Hong, and G.-J. Qi, "Interleaved structured sparse convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8847–8856. 2

[25] B. Graham, "Spatially-sparse convolutional neural networks," *arXiv preprint arXiv:1409.6070*, 2014. 2

[26] A. Morais, J. Egger, and V. Alves, "Automated computer-aided design of cranial implants using a deep volumetric convolutional denoising autoencoder," in *World Conference on Information Systems and Technologies*. Springer, 2019, pp. 151–160. 3

[27] J. Gwak, C. Choy, and S. Savarese, "Generative sparse detection networks for 3d single-shot object detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*. Springer, 2020, pp. 297–313. 4

Appendices

A. Additional Results on skull shape super-resolution.

Figure A.1 shows a visual comparison of shape completion and super-resolution results at the same resolution level.

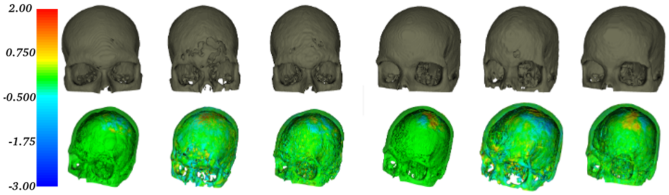

Figure A.1: Comparison of completed skulls at resolution 256. In each example, the first column shows the skull obtained from shape completion at resolution 256. The second and third column show the skull obtained from skull shape super-resolution from 64 and 128. The second row shows the colormap of signed mesh distance between predictions and the ground truth.

B. Implant Generation Results

Figure B.1 shows the generated implants at resolution $512^2 \times Z$ (without any post-processing).

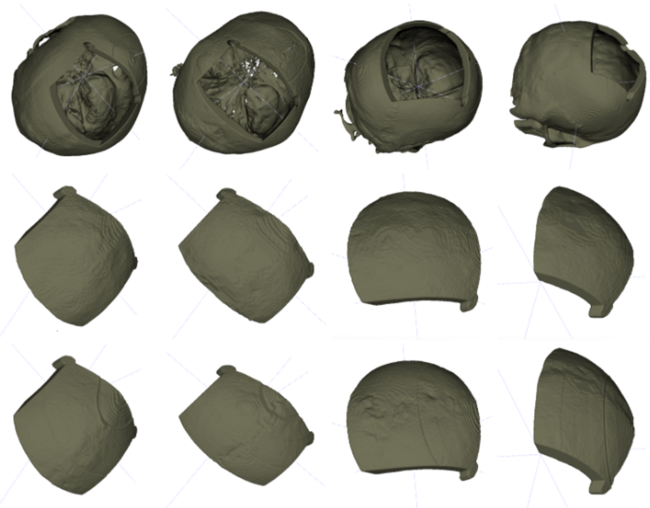

Figure B.1: Implants (second row) obtained by taking the difference between the defective skulls (first row) and the completed skulls at resolution 512. The last row shows the ground truth.

C. Additional Experiments on other Spatially Sparse Medical Images

In this section, additional experiments and results of sparse CNN-based super-resolution on other spatially sparse medical images were provided. The dataset used in the experiments was obtained from the SegTHOR challenge (https://competitions.codalab.org/competitions/21145) that addresses the problem of automatic segmentation of organs at risk. The dataset contains 40 CT scans as well as the segmentation masks of the heart (green), aorta (yellow), trachea (blue) and esophagus (red), as can be seen from Figure C.1. The segmentation masks are spatially sparse with very low voxel occupancy rate (VOR), as can be seen from Table C.1. The dataset contains 20 CT scans without the ground truth segmentation masks for evaluation. The CT scans as well as the segmentation masks are of resolution $512 \times 512 \times Z$.

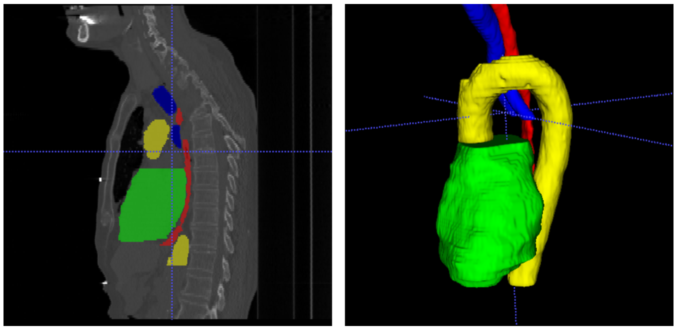

Figure C.1: A CT scan and the ground truth organ segmentation masks of the heart (green), aorta (yellow), trachea (blue) and esophagus (red) from the SegTHOR challenge.

Workflow: Firstly, we downsampled the images to 128^3 and trained a U-Net style dense CNN (1803988 trainable parameters) for automatic segmentation of the organs from the CT scans. Secondly, inference was run on the CT scans in the training and test set to generate the coarse (128^3) segmentation masks. Thirdly, the coarse masks were up-scaled to their original resolution $512 \times 512 \times Z$ via interpolation. Fourthly, we used the up-scaled masks as well as the original ground truth masks from the training set to train a sparse CNN (the same sparse CNN used for skull super-resolution in the main manuscript) for super-resolution. Lastly, we run the inference of the trained sparse CNN on the up-scaled masks from the test set, to obtain the final high-resolution segmentation masks for these organs.

organ	train	test	VOR (%)
aorta	2.05	1.75	0.20
heart	2.46	2.38	0.79
trachea	1.73	1.64	0.04
esophagus	1.77	1.64	0.05

Table C.1: Voxel occupancy rate (VOR) and the memory usage (in *GB*) during training and inference for different organs.

Figure C.2 - C.5 show the qualitative results of the aorta, heart, esophagus and trachea images. It is worth noting that, as the organs in the dataset are even more sparse than the skulls (Table C.1), training on the full $512 \times 512 \times Z$ resolution for the super-resolution task takes only moderate amount of GPU memory (Table C.1), while the super-resolution step can substantially improve the quality of the segmentation masks, as can be seen from Figure C.2 - Figure C.5. Figure C.6 shows the combined segmentation masks of the organs viewed in 2D and 3D, from sparse CNN.

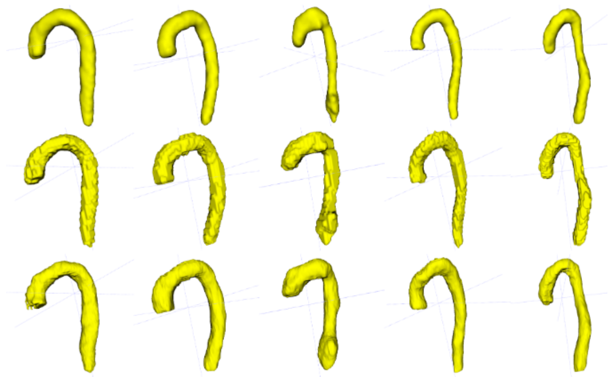

Figure C.2: Super-resolution results of the aorta images. The first to last row shows the coarse aorta mask predictions (128^3) from the dense CNN, the up-scaled aorta masks ($512\times512\times Z$) and the super-resolution output from sparse CNN ($512\times512\times Z$).

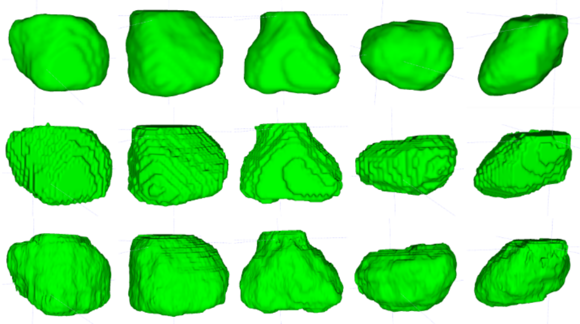

Figure C.3: Super-resolution results of the heart images. The first to last row shows the coarse heart mask predictions (128^3) from the dense CNN, the up-scaled heart masks ($512\times512\times Z$) and the super-resolution output from sparse CNN ($512\times512\times Z$).

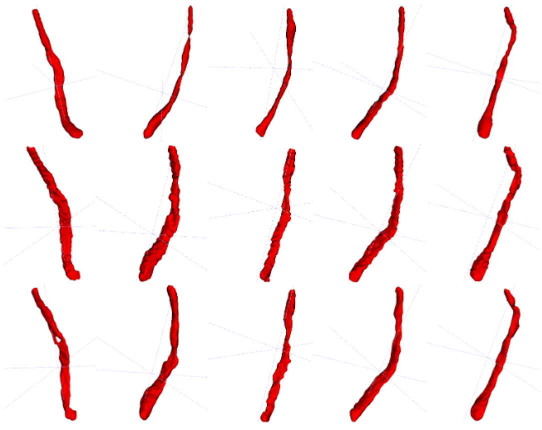

Figure C.4: Super-resolution results of the esophagus images. The first to last row shows the coarse esophagus mask predictions (128^3) from the dense CNN, the up-scaled esophagus masks ($512 \times 512 \times Z$) and the super-resolution output from sparse CNN ($512 \times 512 \times Z$).

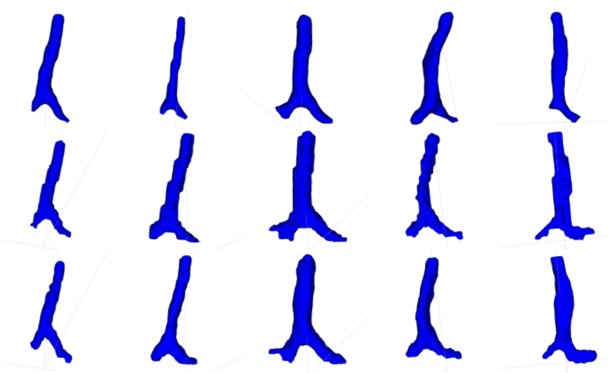

Figure C.5: Super-resolution results of the trachea images. The first to last row shows the coarse trachea mask predictions (128^3) from the dense CNN, the up-scaled trachea masks ($512 \times 512 \times Z$) and the super-resolution output from sparse CNN ($512 \times 512 \times Z$).

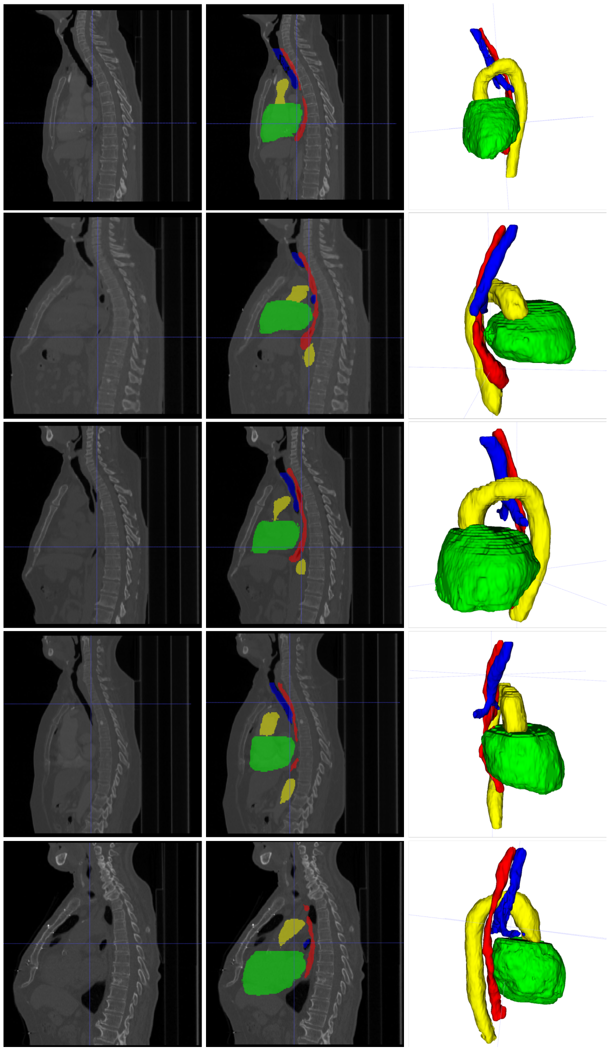

Figure C.6: The segmentation masks of the organs viewed in 2D (second column) and 3D (third column). The first column shows a slice of the CT scan.