Sparse Convolutional Neural Networks for Medical Image Analysis

Jianning Li

Background & Motivation

- Medical Images are Large and modern GPUs are still memory-constrained

  . typical resolution: $512 \times 512 \times Z$

  . typical desktop GPU memory: 6 ~ 32 *GB*

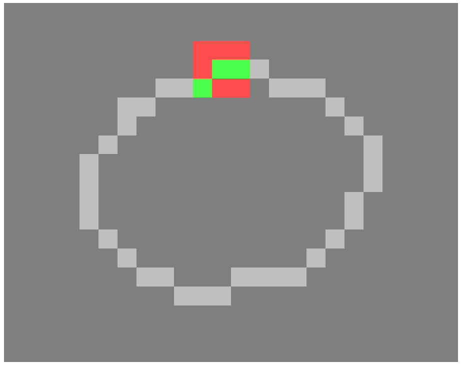
- Some medical Images are spatially sparse with very low voxel occupancy rate (VOR)

  . skull: up to approx. 10%

  . segmentation masks of human organs: as low as 0.04%

  . dense convolution: number of non-empty points grow rapidly with each layer [1]

- Convolutions that operate only on non-empty voxels?



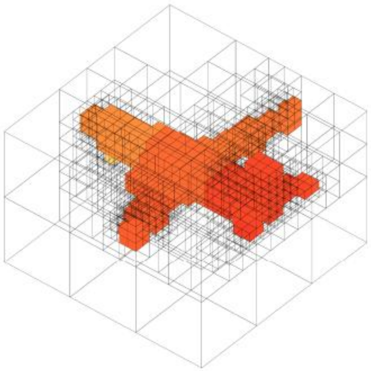
Source: https://github.com/facebookresearch/SparseConvNet

[1] Graham, B. and van der Maaten, L., 2017. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*.

Background & Motivation: Sparse CNN

- Sparse CNN for spatially sparse data
  . octree: O-CNN [1], OctNet [2]
  . coordinates and features [4]

➡ to get rid of the dominant empty points that do not carry valid information of the target



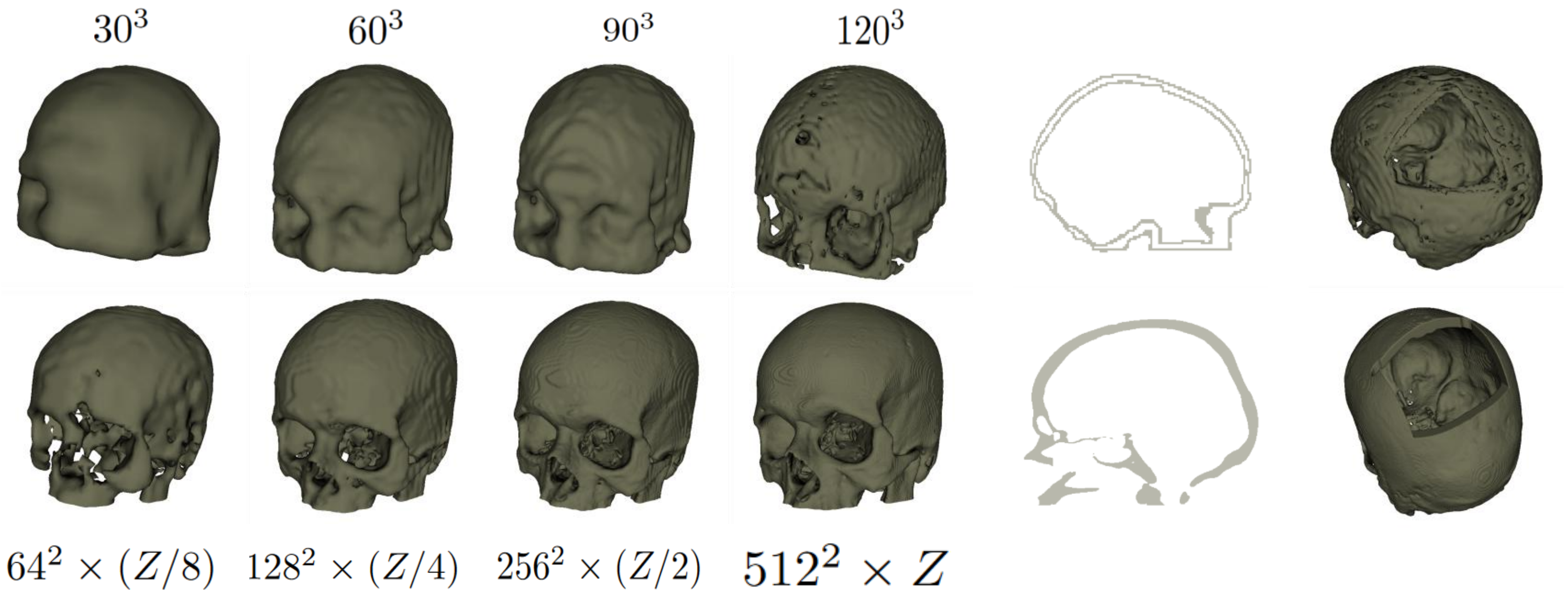
compact octree representation of 3D shapes (source: https://griegler.github.io/papers/octnet_slides.pdf)

- CNN with sparse parameters
  . parameters are mostly zero after pruning
  . densely trained parameters have a lot of redundancy [3]
  . increase parameter sparsity without substantially decrease in accuracy

[1] Wang, P.S., et al. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. In TOG 2017.
[2] Riegler, G., et al. Octnet: Learning deep 3d representations at high resolutions. In CVPR 2017.
[3] Liu, B.et al. Sparse convolutional neural networks. In CVPR 2015.
[4] Choy, C., et al. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In CVPR 2019.

Background & Motivation: Skull reconstruction

- Skull images are large $(512 \times 512 \times Z)$, binary and spatially sparse (VOR ~10%)
  . MRI (top row) and CT (bottom row) skull images at different resolutions

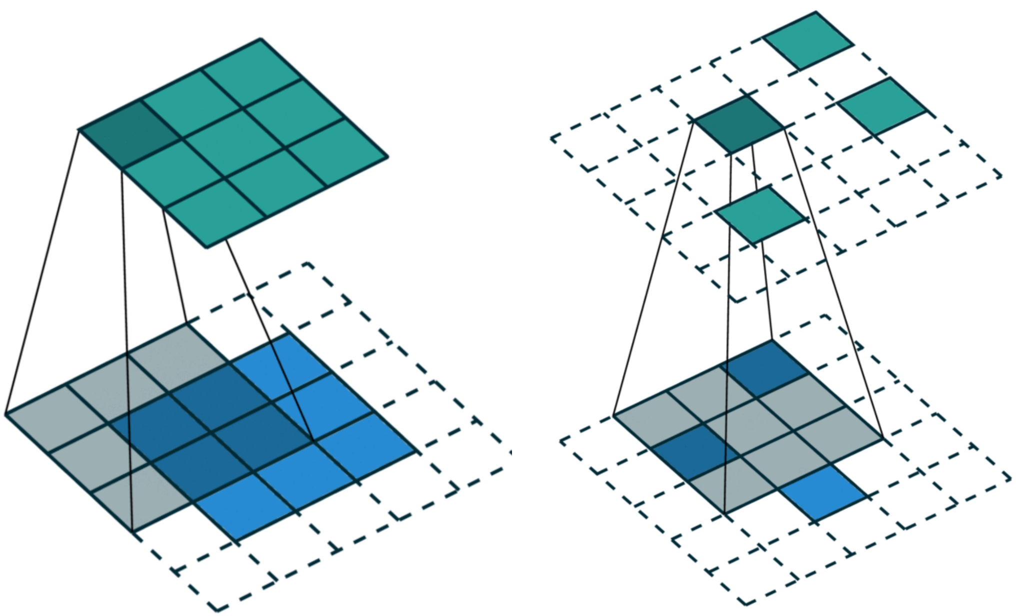


Skull shape completion: automatically complete an incomplete skull
Skull shape-super-resolution: given a coarse skull, reconstruct a high-resolution skull

Method

Minkowski Engine

- Convolution defined on specified points (left) instead of on the entire voxel grid (right)



Dense convolution

Sparse convolution

Source: https://nvidia.github.io/MinkowskiEngine/overview.html

for sparse binary volumes of static data:

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

coordinates of non-zero voxels:

$$\mathcal{C}_{in} \in \mathbb{Z}^{N \times 3}$$

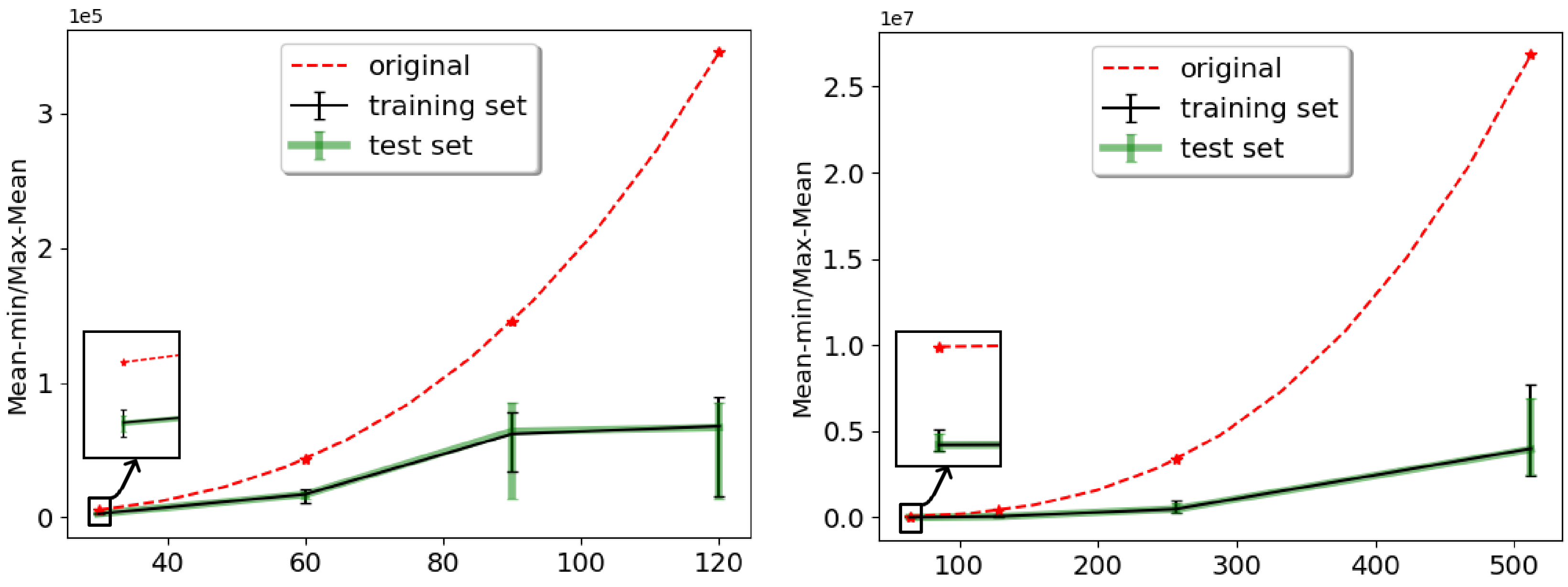
associated feature vectors (voxel values):

$$\mathcal{F}_{in} \in \mathbb{R}^{N \times 1}$$

Method

Minkowski Engine

- **Convolution defined on non-empty points**: comparison of the non-empty voxel number and total voxel number on the skull datasets



MRI skull dataset CT skull dataset

. Overall data memory occupancy (y-axis) grows cubically wrt. Image resolution (x-axis)
. Binary skull images were stored as *int8* (MRI) and *int32* (CT)

Method

Minkowski Engine

- **Convolution defined on non-empty points:** comparison of memory usage and computational complexity (floating point operations)

training:

- Input and ground truth image batches
- Intermediate layers‘ output
- Network parameters
- Back-propogation: errors, gradients
- Optimizers

inference:

- Input image batches
- Intermediate layers‘ output
- Network parameters

1. Output size of the intermediate layer i: $N_{fi} = \frac{1}{s}(N_{fi-1} + 2p - Ks)$

   s, p, Ks: size of stride, padding and kernel

2. Floating point operations: product of Nfi, Ks and in and out number of channels

Overall GPU memory usage measurement: query GPU memory occupancy at 50-millisecond intervals for N_train epochs (batch size=1)

Method

Minkowski Engine

- Sparse CNN for shape completion and super-resolution

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	

* stride 2
bold: transposed generative layers

- Auto-encoder architecture
- Sparse convolutional layers
- ch is the list of channel numbers of each layer

convolutions at coordinate D:

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

feature vector at coordinate D:

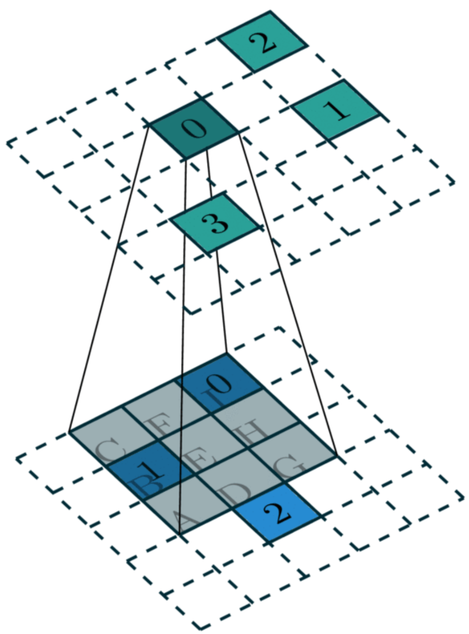
$$\mathcal{F}_{in}^{i} \in \mathbb{R}^{C_{i-1}^{out}} \quad D \in \mathbb{Z}^3$$

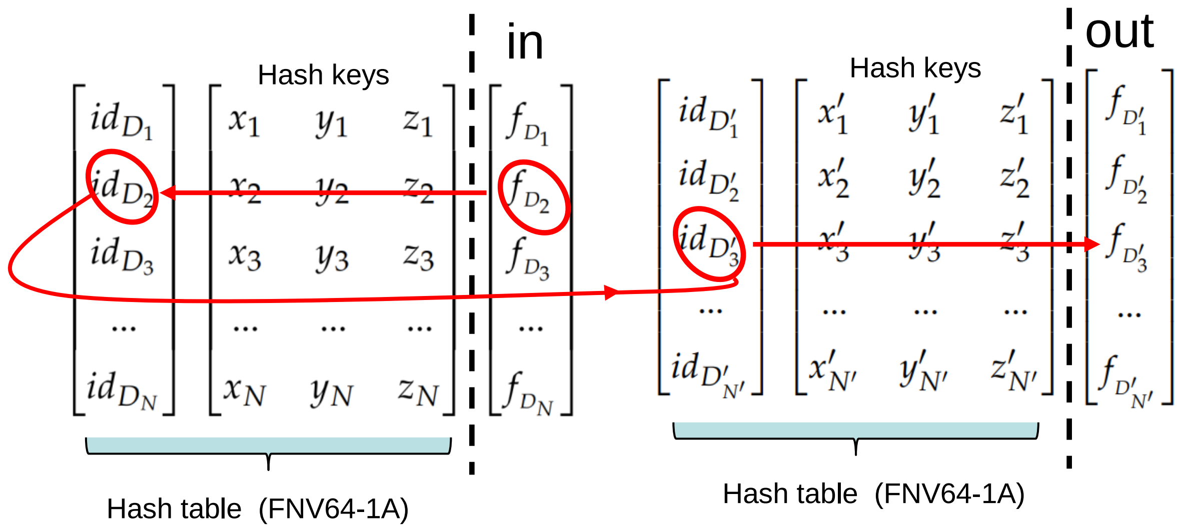
Method

Minkowski Engine

- Coordinate mapping

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





Algorithm 2 Generalized Sparse Convolution

Require: Kernel weights $\mathbf{W}$, input features F^i, output feature placeholder F^o, convolution mapping $\mathbf{M}$,

1: $F^o \leftarrow \mathbf{0}$ // set to 0

2: **for all** $W_{\mathrm{i}}, (I_{\mathrm{i}}, O_{\mathrm{i}}) \in (\mathbf{W}, \mathbf{M})$ **do**

3: $\quad F_{\mathrm{tmp}} \leftarrow W_{\mathrm{i}}[F^i_{I_{\mathrm{i}}[1]}, F^i_{I_{\mathrm{i}}[2]}, \ldots, F^i_{I_{\mathrm{i}}[n]}]$ // (cu)BLAS

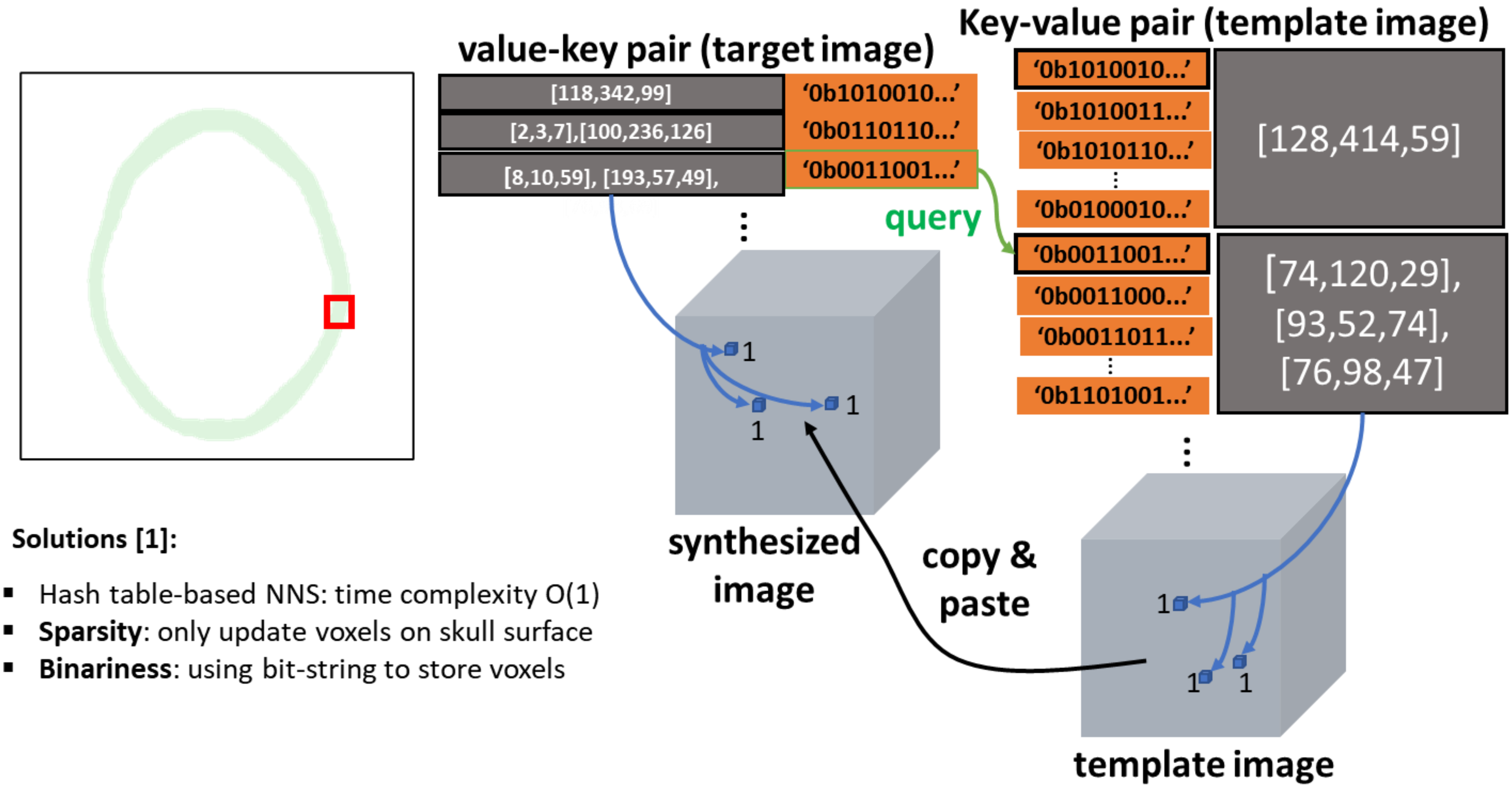
4: $\quad F_{\mathrm{tmp}} \leftarrow F_{\mathrm{tmp}} + [F^o_{O_{\mathrm{i}}[1]}, F^o_{O_{\mathrm{i}}[2]}, \ldots, F^o_{O_{\mathrm{i}}[n]}]$

5: $\quad [F^o_{O_{\mathrm{i}}[1]}, F^o_{O_{\mathrm{i}}[2]}, \ldots, F^o_{O_{\mathrm{i}}[n]}] \leftarrow F_{\mathrm{tmp}}$

6: **end for**

Source: https://nvidia.github.io/MinkowskiEngine/overview.html

Method



Solutions [1]:

- Hash table-based NNS: time complexity O(1)
- **Sparsity**: only update voxels on skull surface
- **Binariness**: using bit-string to store voxels

[1] Li, J., Pepe, A., Gsaxner, C., Jin, Y. and Egger, J., 2021. Learning to Rearrange Voxels in Binary Segmentation Masks for Smooth Manifold Triangulation. *arXiv preprint arXiv:2108.05269*.

Method

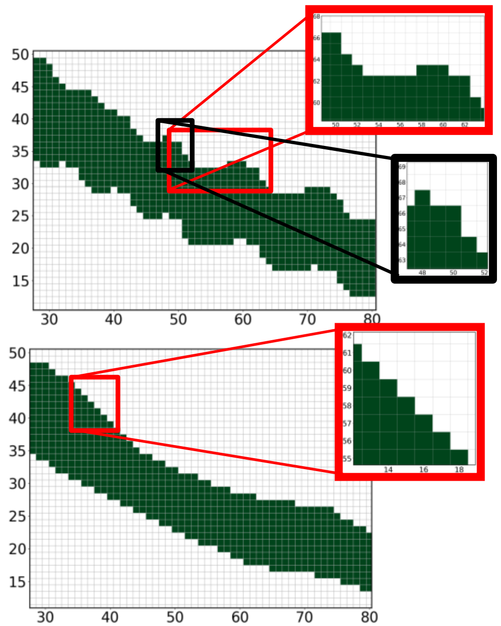
Minkowski Engine

- Sparse CNN shape completion (sc) and super-resolution (sr)

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

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

transposed generative convolution [1]



$$\mathcal{C}^{sr}_{out} = \begin{bmatrix} x'_1 & y'_1 & z'_1 \\ x'_2 & y'_2 & z'_2 \\ \dots & \dots & \dots \\ x'_N & y'_N & z'_N \end{bmatrix}, \mathcal{F}^{sr}_{out} = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix}$$

$$\mathcal{C}^{sr}_{out} \cap \mathcal{C}_{in} \neq \emptyset$$

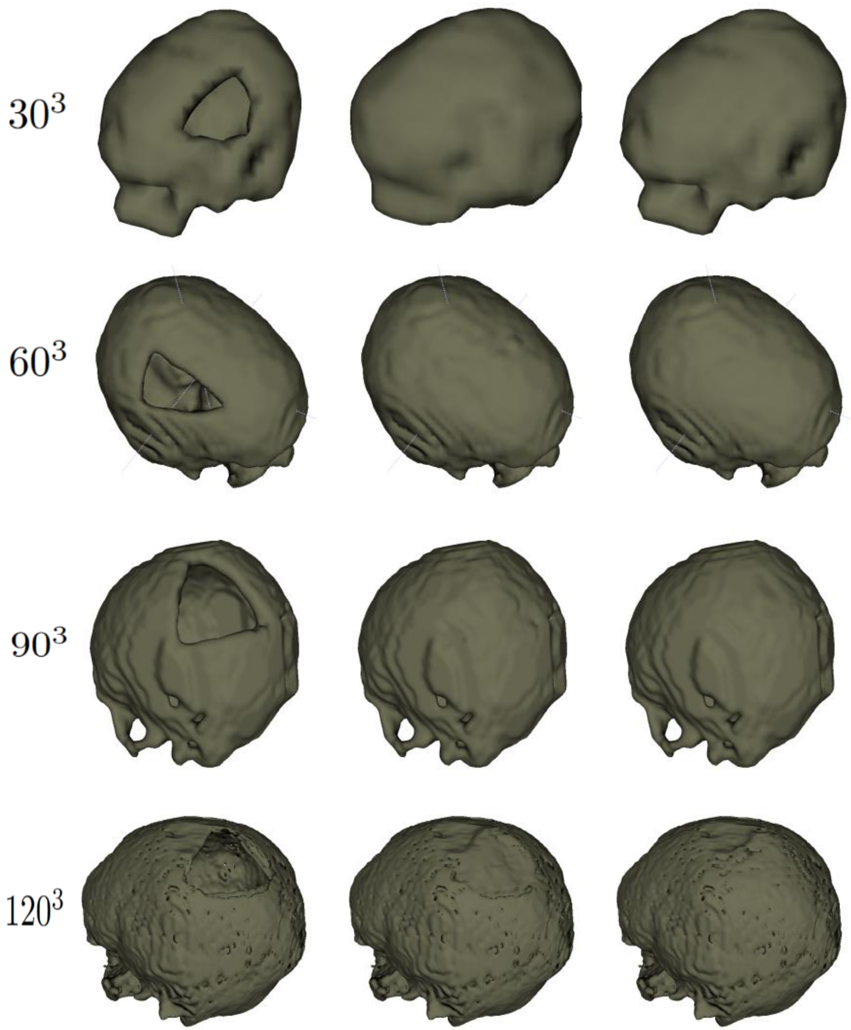
non-zero voxels partially rearranged [2]

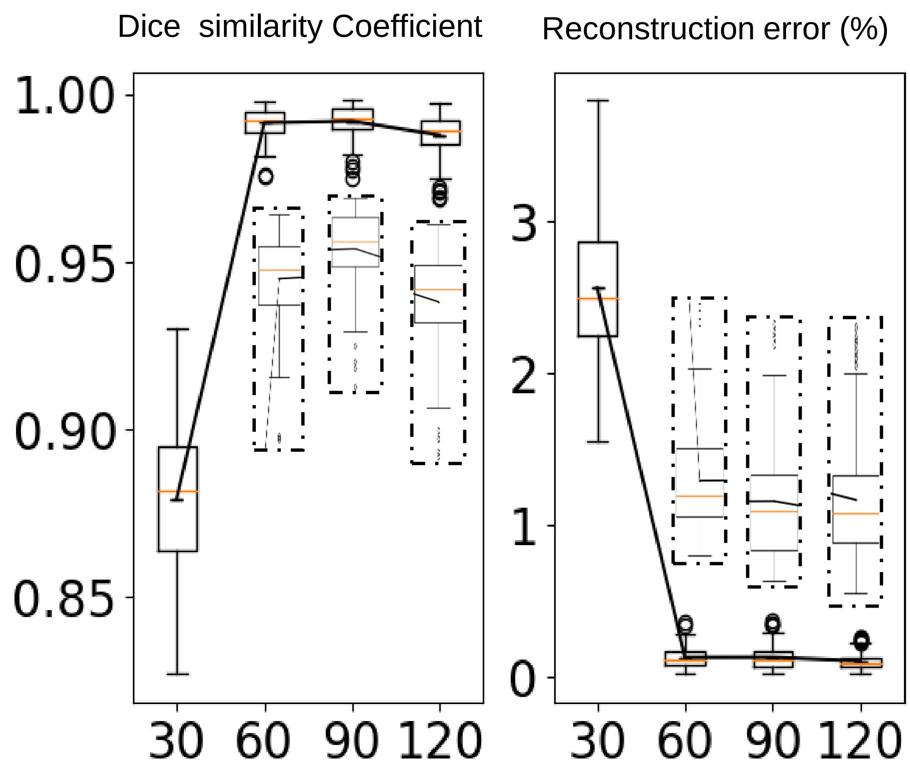
[1] Gwak, J., Choy, C. et al. Generative sparse detection networks for 3d single-shot object detection. In ECCV 2020

[2] Li, J., Pepe, A., et al. Learning to Rearrange Voxels in Binary Segmentation Masks for Smooth Manifold Triangulation. arXiv 2021.

Experiments & Results

- Shape completion on the MRI skull dataset

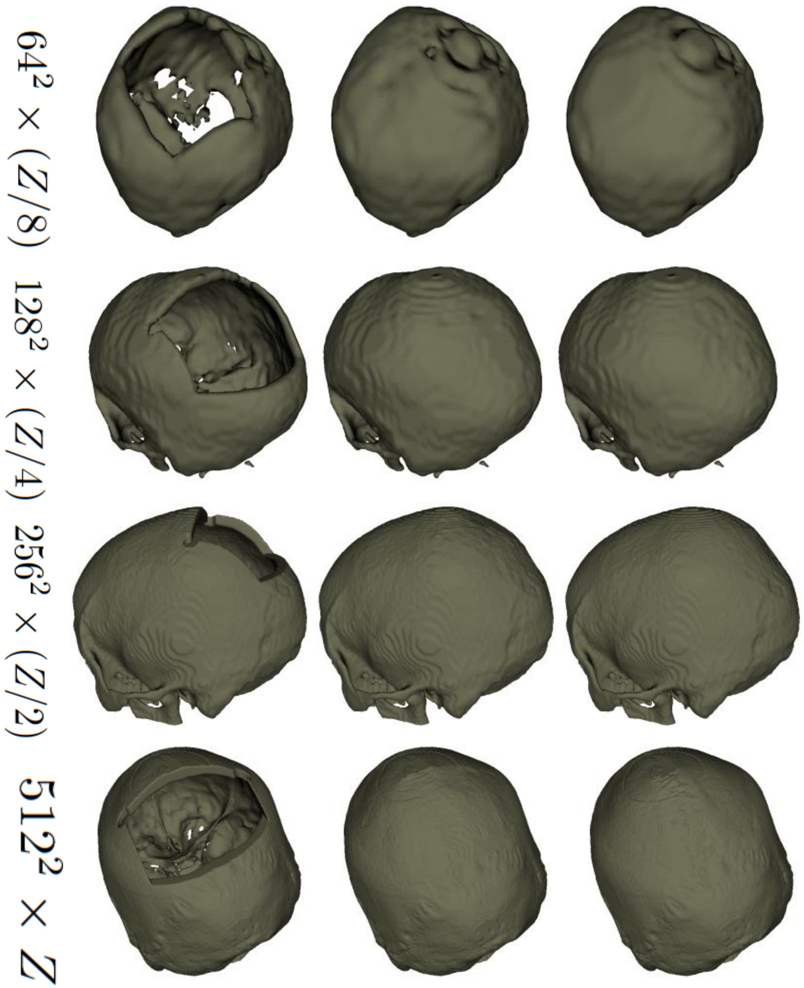


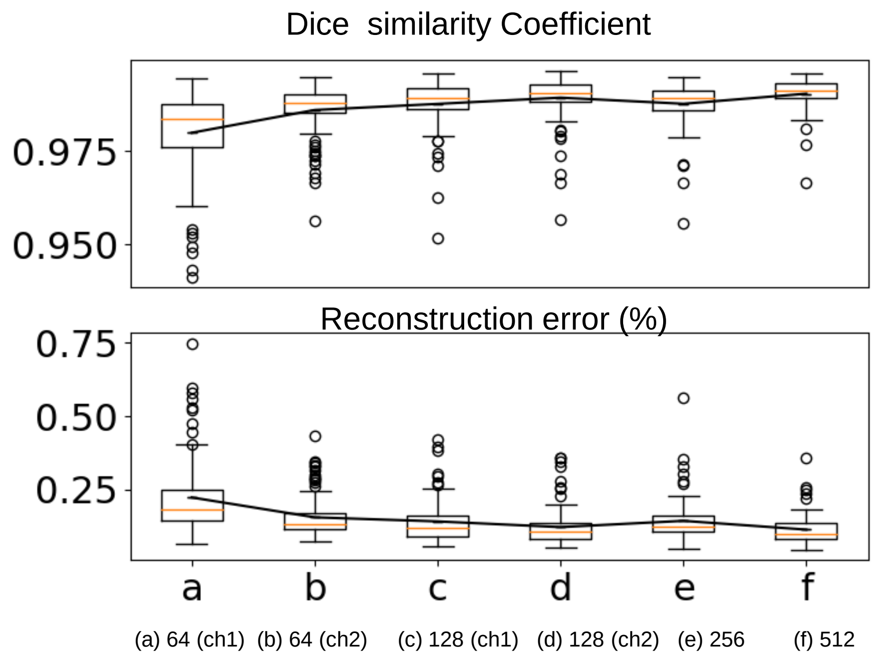


Experiments & Results

- Shape completion on the CT skull dataset

ch1 (0.435M params)
ch2 (18.14M params)

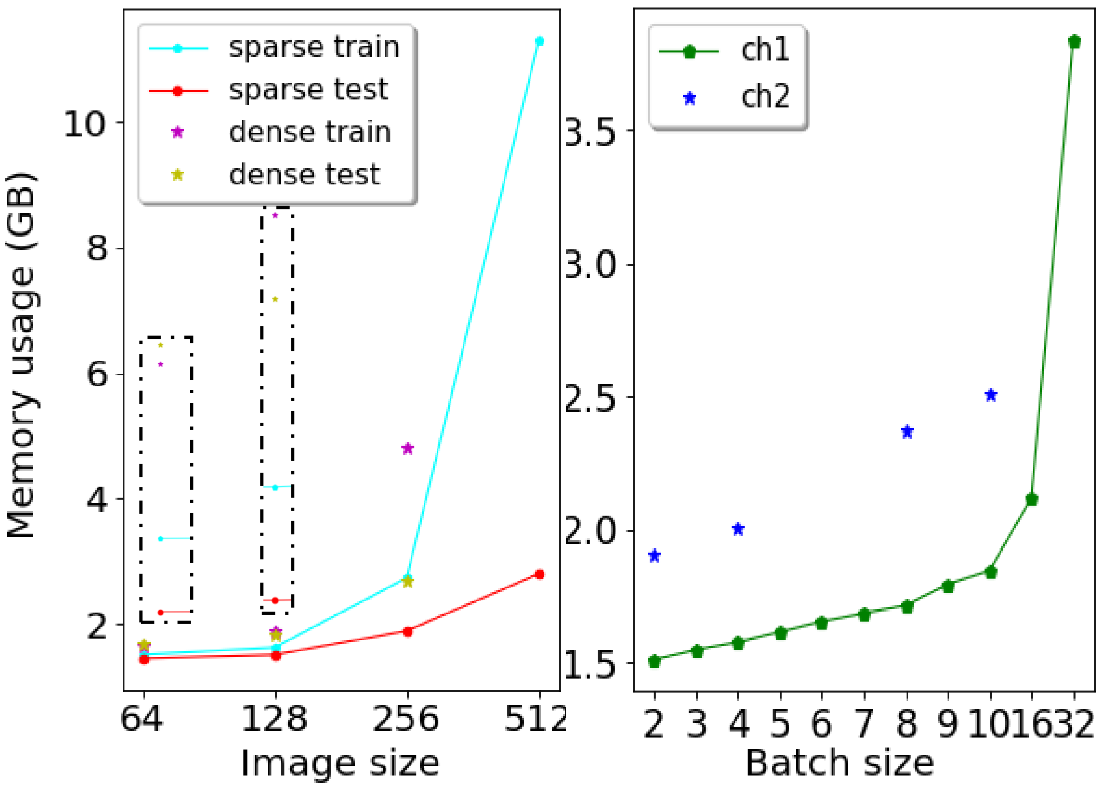




- 0.9903 DSC – Current state of the art in skull shape completion!
- increasing model complexity increases prediction accuracy.

Experiments & Results

- Memory usage comparison



ch1 (0.435M params)
ch2 (18.14M params)

Memory wrt. resolution

cat. \ 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	-

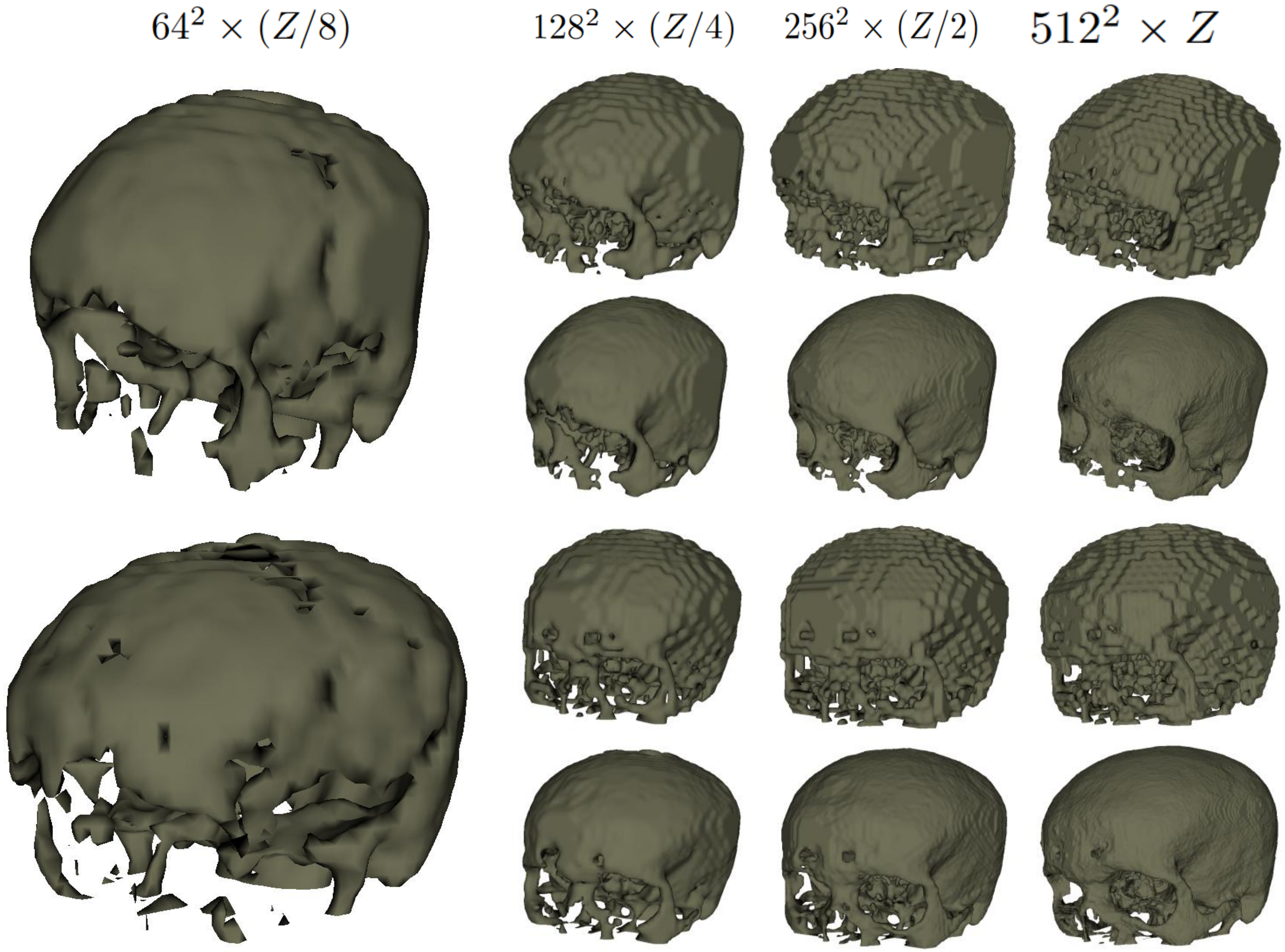
Memory wrt. batch size

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	-	-

- sparse CNN inference: memory usage grows linearly wrt. Image res.
- sparse CNN training: memory usage grows linearly at res. 256 and below and subquadruply at res. 512
- x40 increase in parameters leads to less than x2 memory usage for sparse CNN
- training sparse CNN at full resolution is reasonably fast, in contrast to dense CNN.

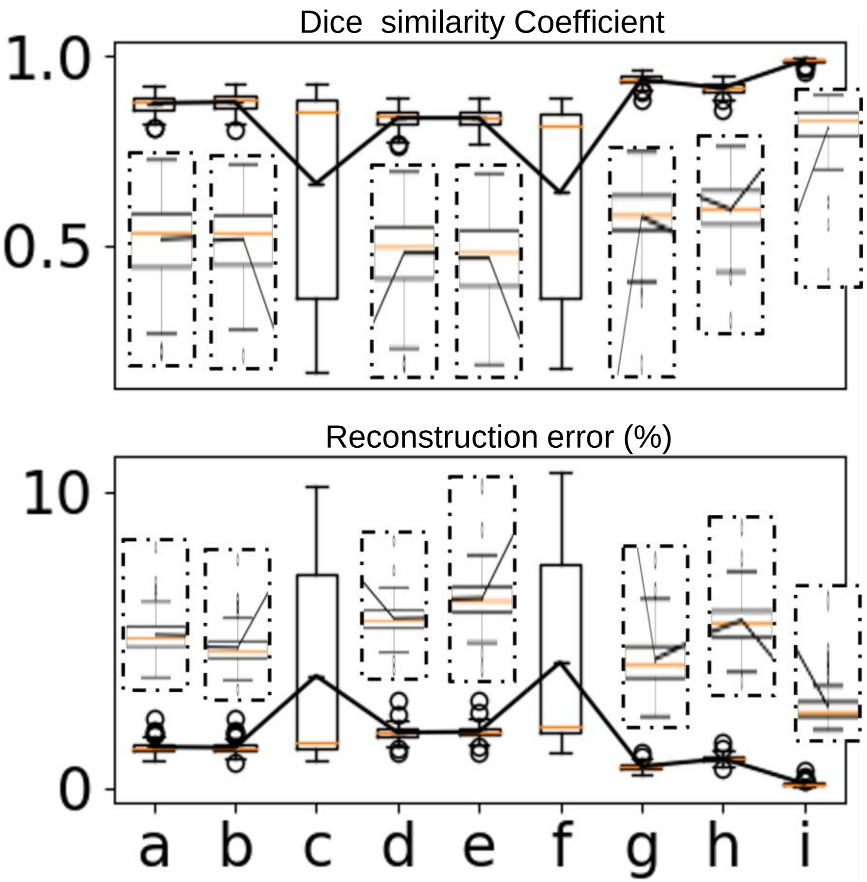
Experiments & Results

- Super-resolution on the CT skull dataset

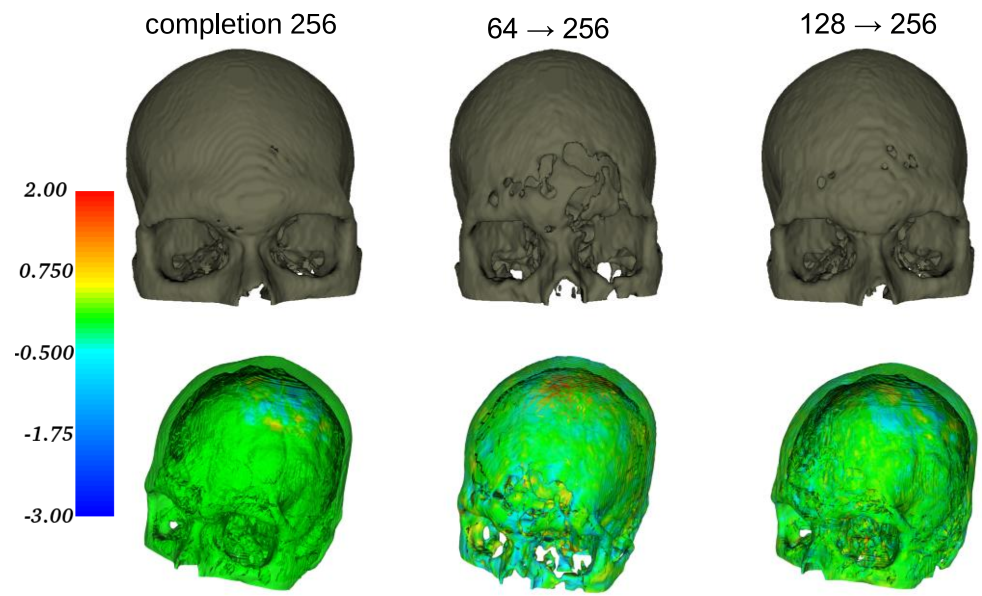


Experiments & Results

- Super-resolution on the CT skull 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. → super-resolution ⇒ interpolation

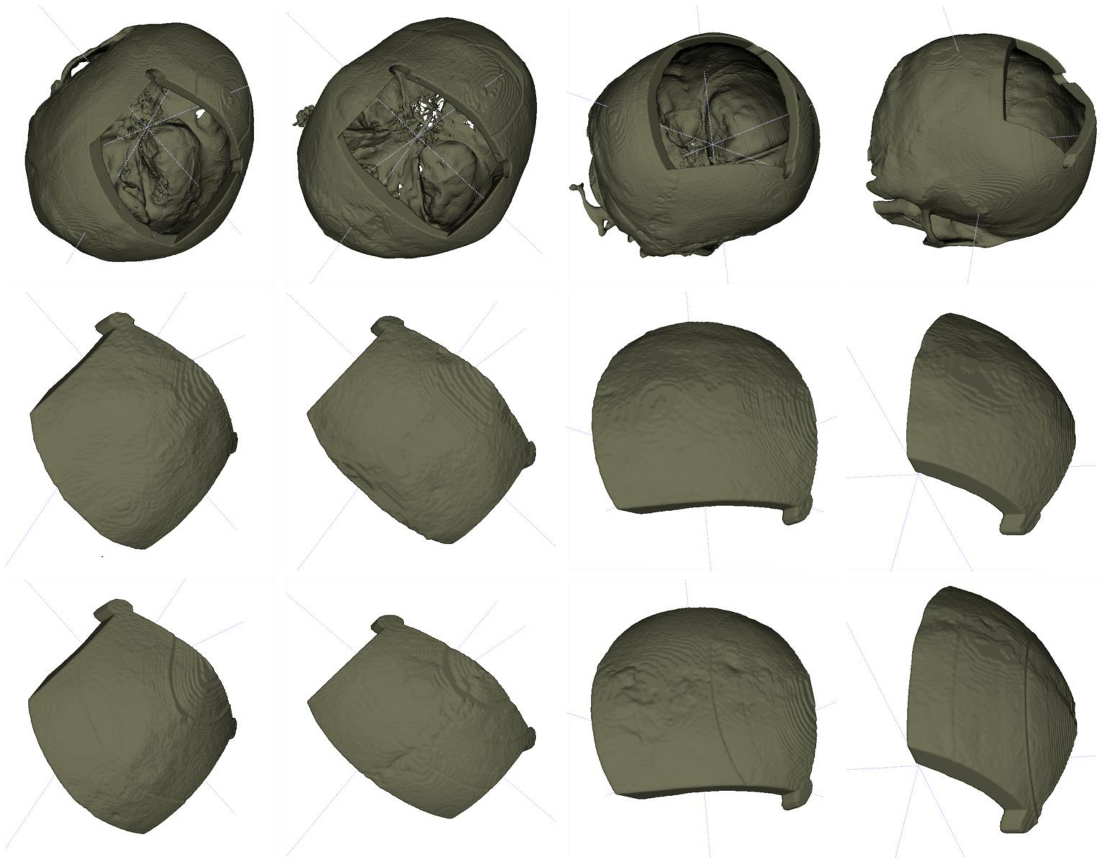


signed distance wrt. gt

- the sparse CNN model is better at the shape completion task
- super-resolution with smaller gaps has better results

Experiments & Results

- Implant generation at resolution 512x512xZ

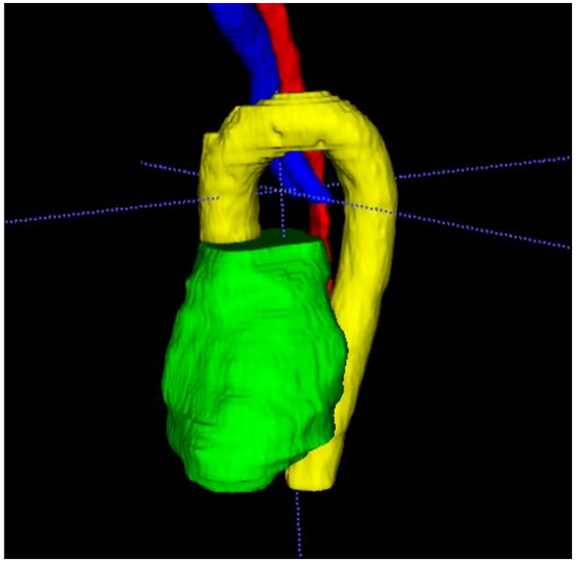


- [1] ensemble of four models
- two V100 GPUs (32 GB each)
- downsampled
- massive data augmentation: 10000 training samples

[1] Ellis, D.G. and Aizenberg, M.R., 2020, October. Deep Learning Using Augmentation via Registration: 1st Place Solution to the AutoImplant 2020 Challenge. In Cranial Implant Design Challenge (pp. 47-55). Springer, Cham.

Experiments & Results

- Segmentation of sparse medical images



heart (green), aorta(yellow), trachea (blue) and esophagus (red) from the SegTHOR challenge (https://competitions.codalab.org/competitions/21145)

- resolution: 512x512xZ
- workflow: dense CNN segmentation (128^3) – sparse CNN super-resolution (512x512xZ)
- organ masks voxel occupancy rates are very low

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 S1.Voxel occupancy rate (VOR) and the memory usage (in GB) during training and inference for different organs.

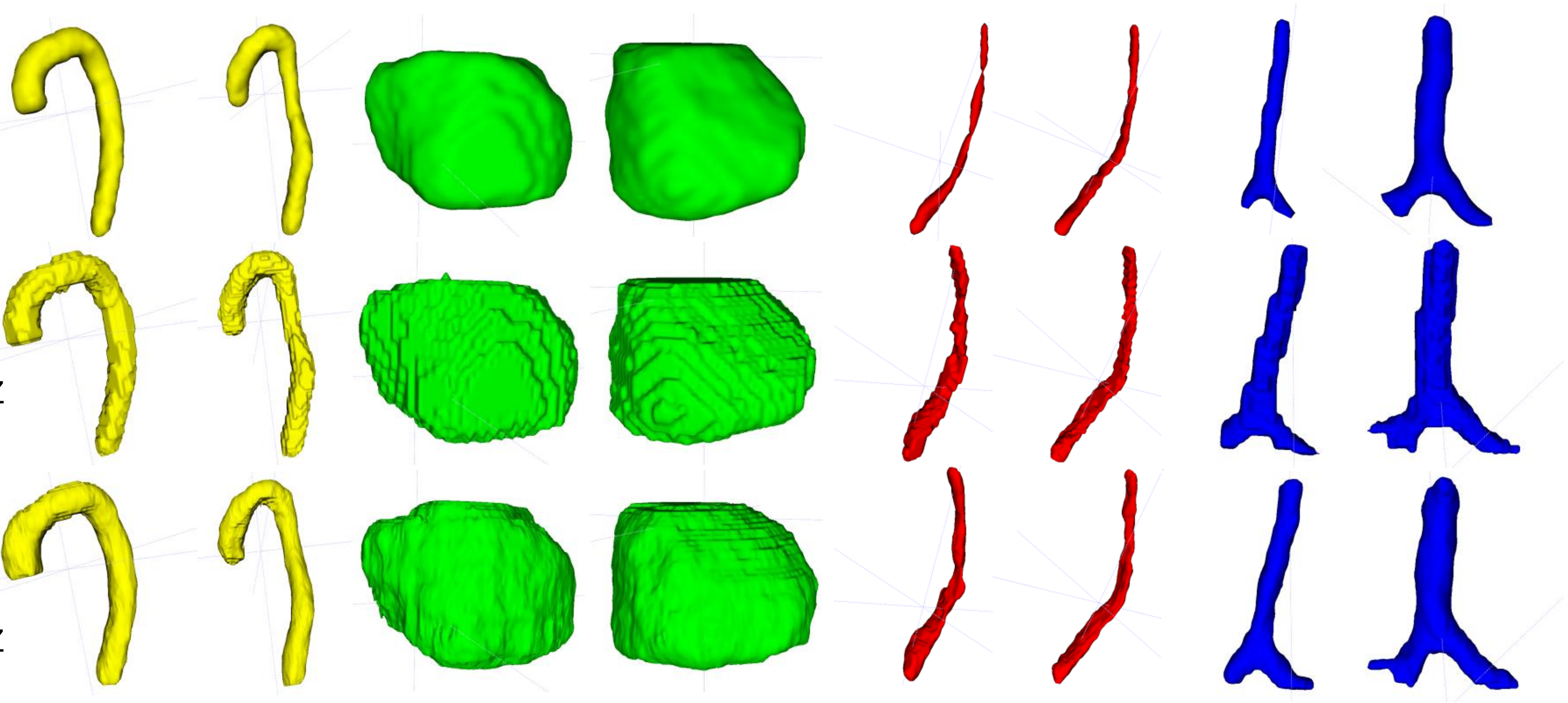
Experiments & Results

- Segmentation of sparse medical images

128^3

512^2xZ

512^2xZ



Conclusions

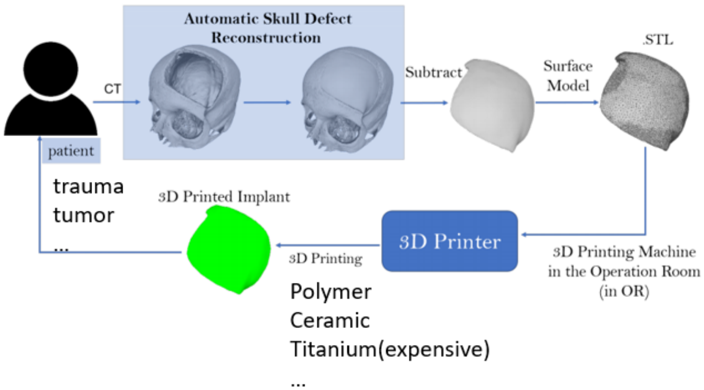
- Sparse CNN outperforms dense CNN wrt. speed, performance, memory and computation efficiency, on sparse problems

- Minkowski Engine (ME) was a general-purpose library capable of processing 4D spatio-temporal data. We have showed its applicability on sparse binary volumes of static data (skulls, organ masks, etc), on different medical image analysis tasks

- In ME or other sparse CNN libraries/methods, voxel coordinates are involved in convolution computations. Hash table is generally used to prevent querying the coordinates from slowing down convolutions

Kroviakov, A., Li, J. and Egger, J., 2021, October. *Sparse Convolutional Neural Network for Skull Reconstruction*. In Cranial Implant Design Challenge (pp. 80-94). Springer, Cham.

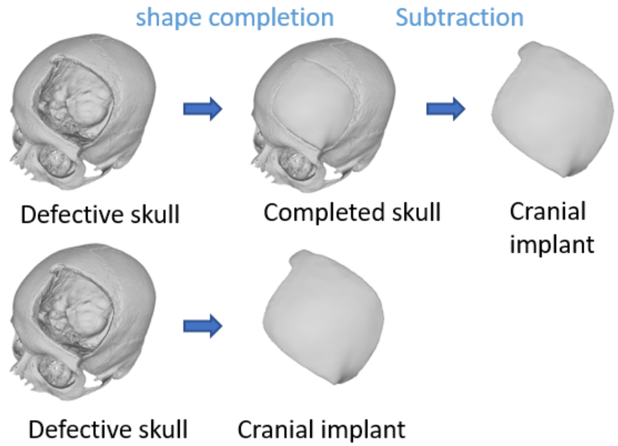
Automated implant design



In comparison to current practice of cranial implant design:

- low cost & fast
- in operation room (in-OR) design & manufacturing
- no secondary surgeries required (cranioplasty can be performed hours after craniectomy)

Formulation



- 3D shape completion, shape learning and modeling
- CNN, classic image processing

Problems

- ○ <u>Generalizability</u>
  - generalize to various defect shapes
  - generalize to various skull shapes
  - generalize to clinical cases
- ○ <u>Sparse problems</u>
  - skull image are large (512*512*Z)
  - desktop GPU memory is limited
  - training is slow
- ○ <u>Clinical utility</u>
  - transfer synthetically trained model to clinical data
  - quantify experts' evaluation criteria
  - user interface (https://www.youtube.com/watch?v=pt-jw8nXzgs)

Jianning.Li@uk-essen.de

Thank You