Australia's National Science Agency

Cortical Surface Retrieval via Deformable Models

A Vacation Student Project

Darren Fu, Rodrigo Santa Cruz, Leo Lebrat, Jurgen Fripp, and Olivier Salvado

d.fu@uq.net.au

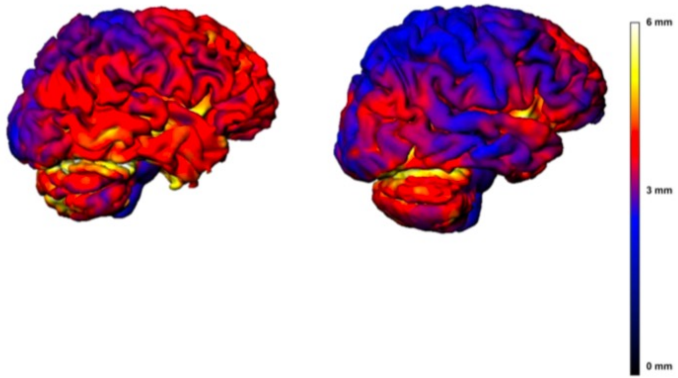
THE AUSTRALIAN E-HEALTH RESEARCH CENTRE

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

QUT Queensland University of Technology

PROBLEM STATEMENT

Numerous neurodegenerative diseases affect the structure of our brain



Measure of thickness for AD (Alzheimer's Disease) and HC (healthy control).

PROBLEM STATEMENT

Cortical thickness measurements allows the diagnosis of:

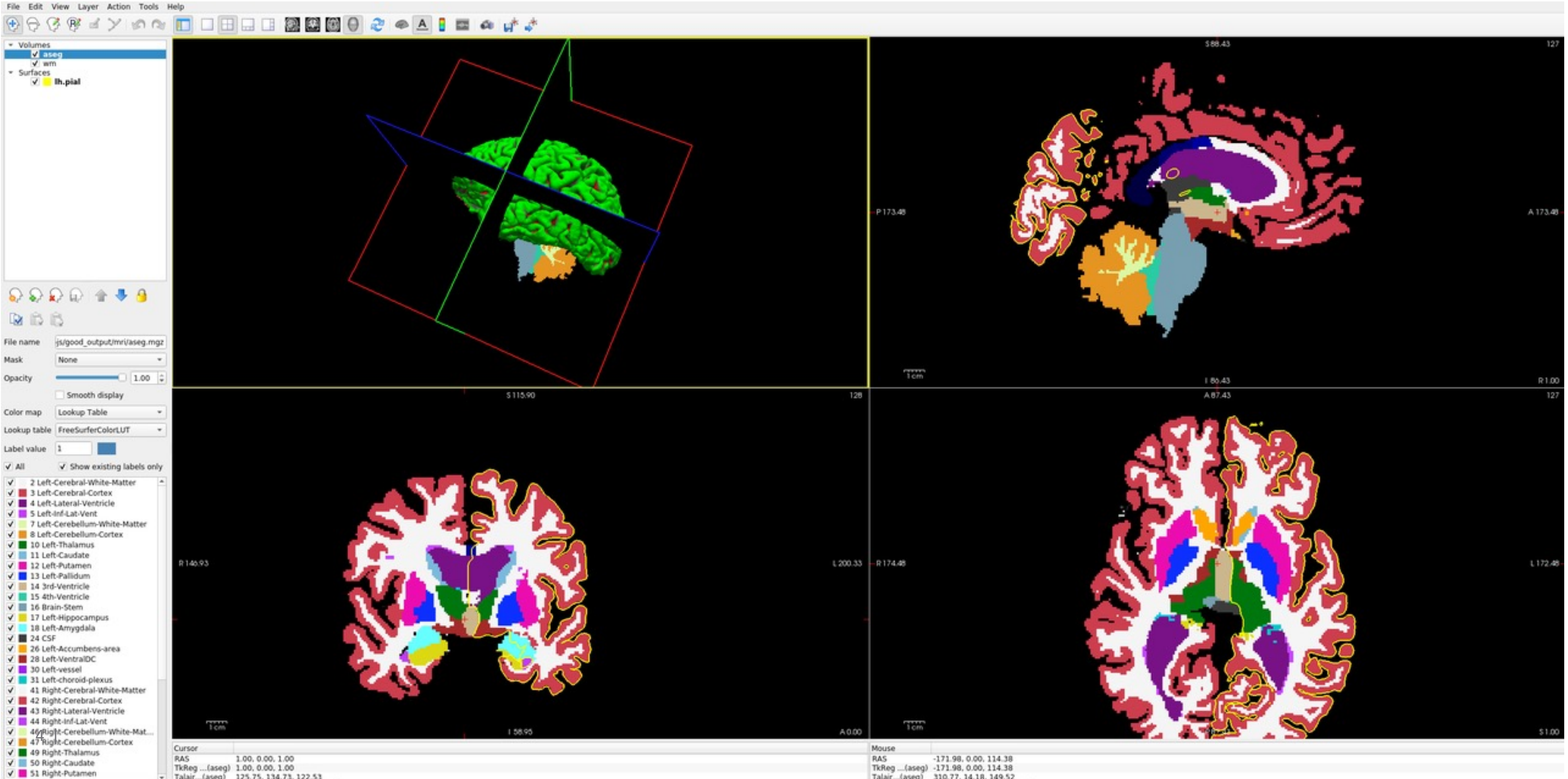
- Alzheimer's disease
- Parkinson's disease
- …

Voxel Image

3D Mesh

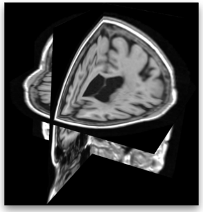
EXISTING METHODS

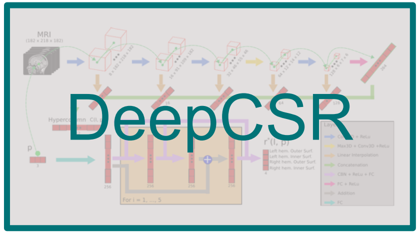
FreeSurfer



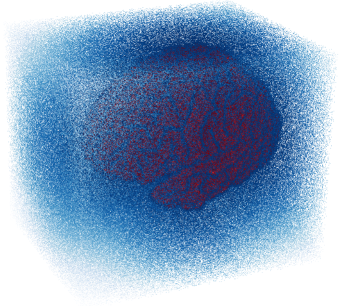
EXISTING METHODS

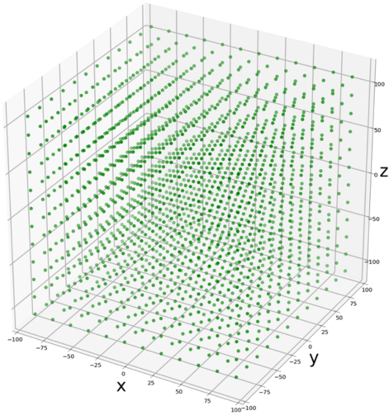
Input Image



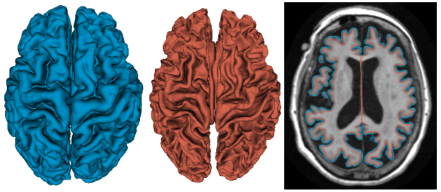


Implicit Surface Representation



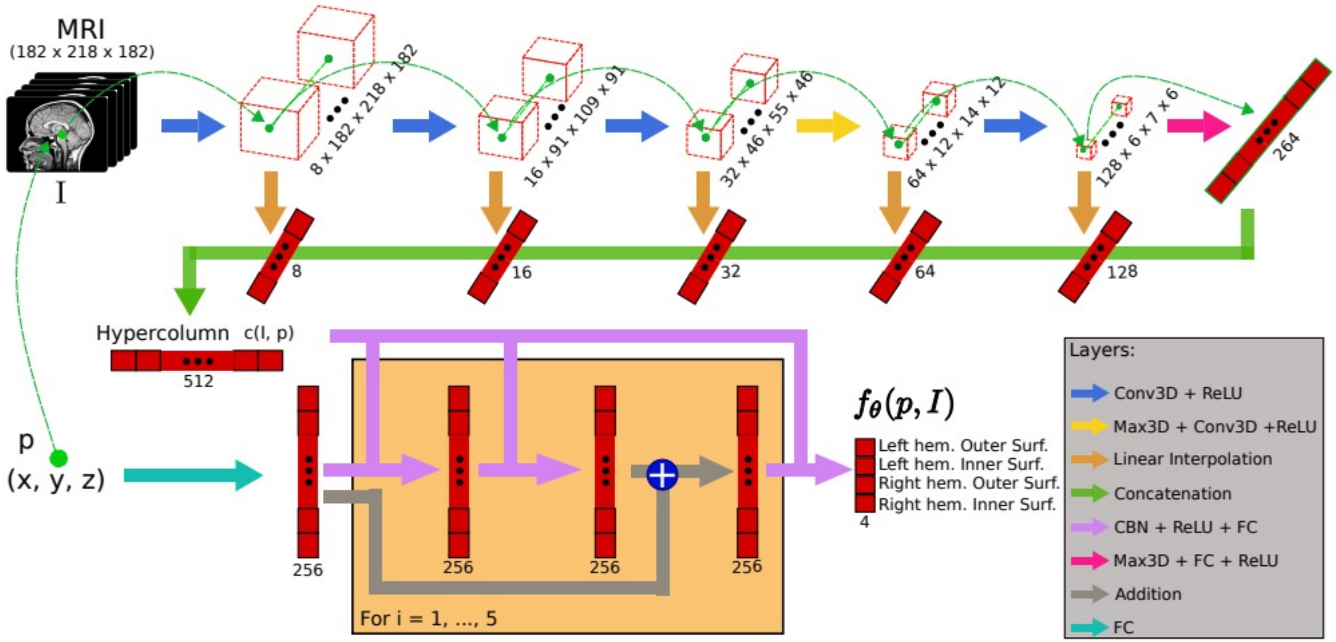


x_0	y_0	z_0
x_1	y_1	z_1
⋮	⋮	⋮
x_n	y_n	z_n

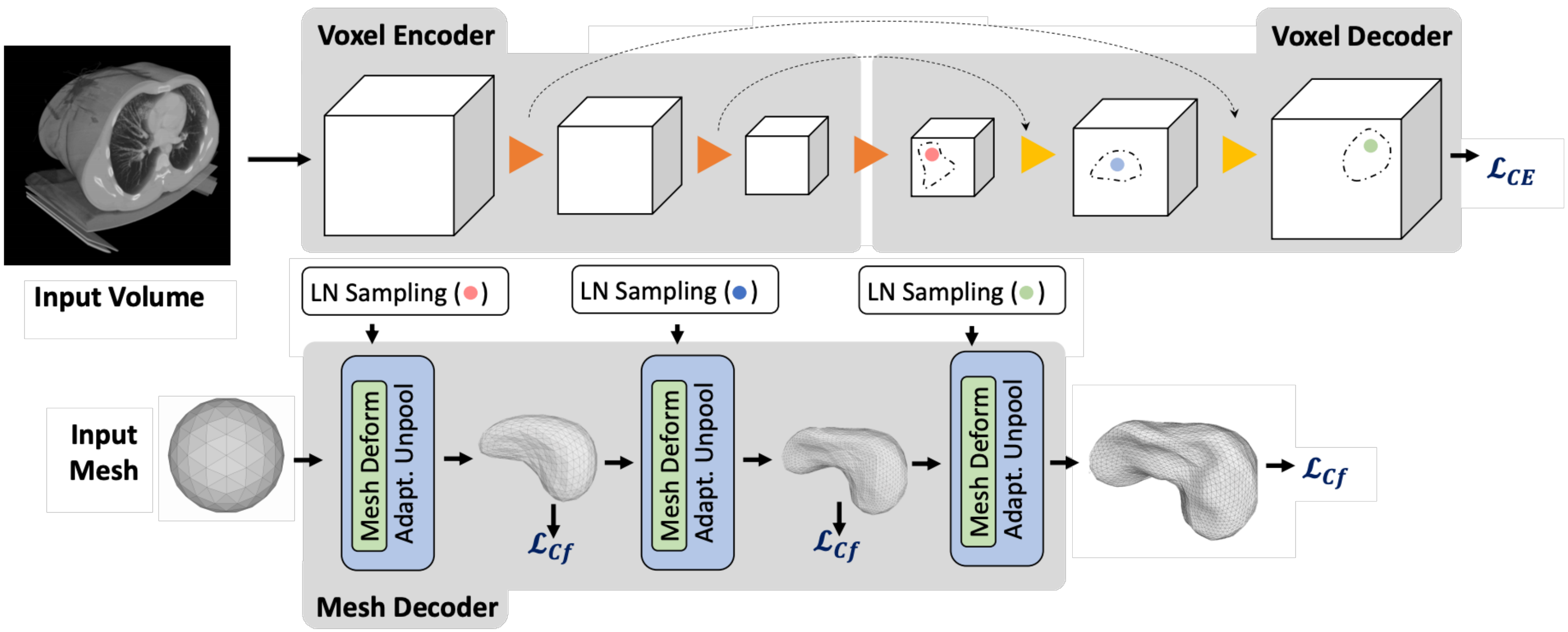


Mesh Prediction

EXISTING METHODS

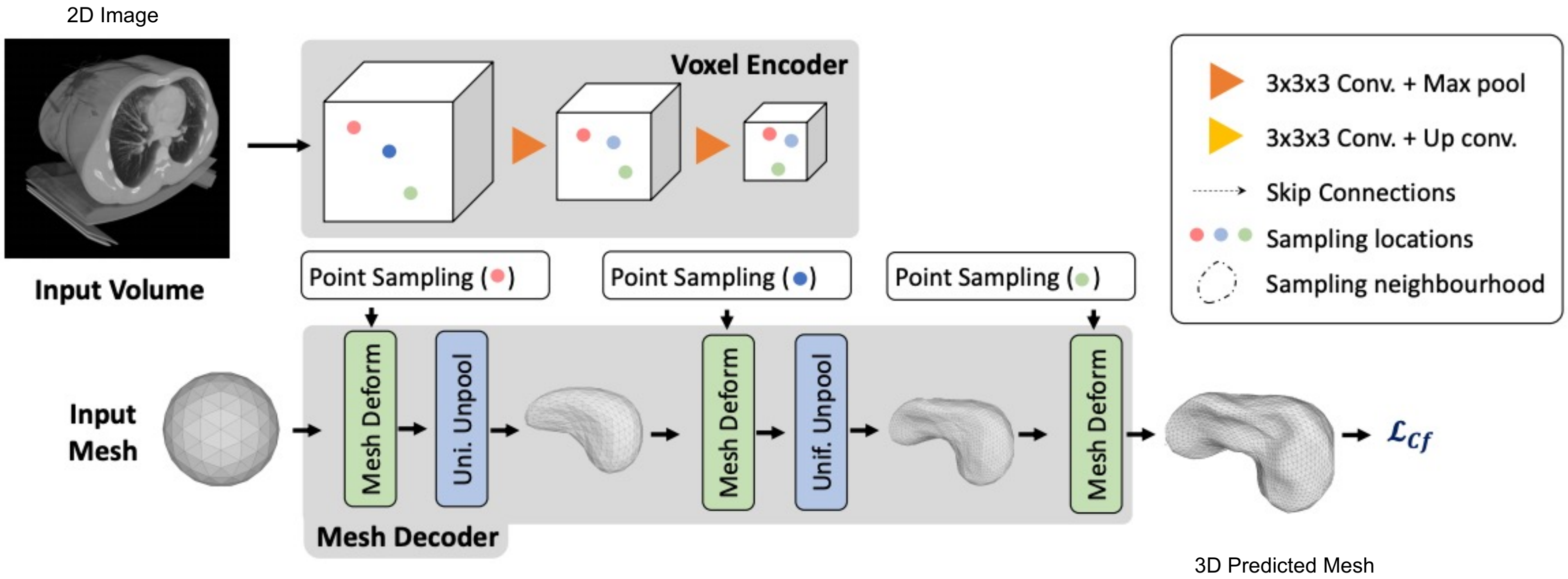


EXISTING METHODS



Voxel2Mesh

EXISTING METHODS

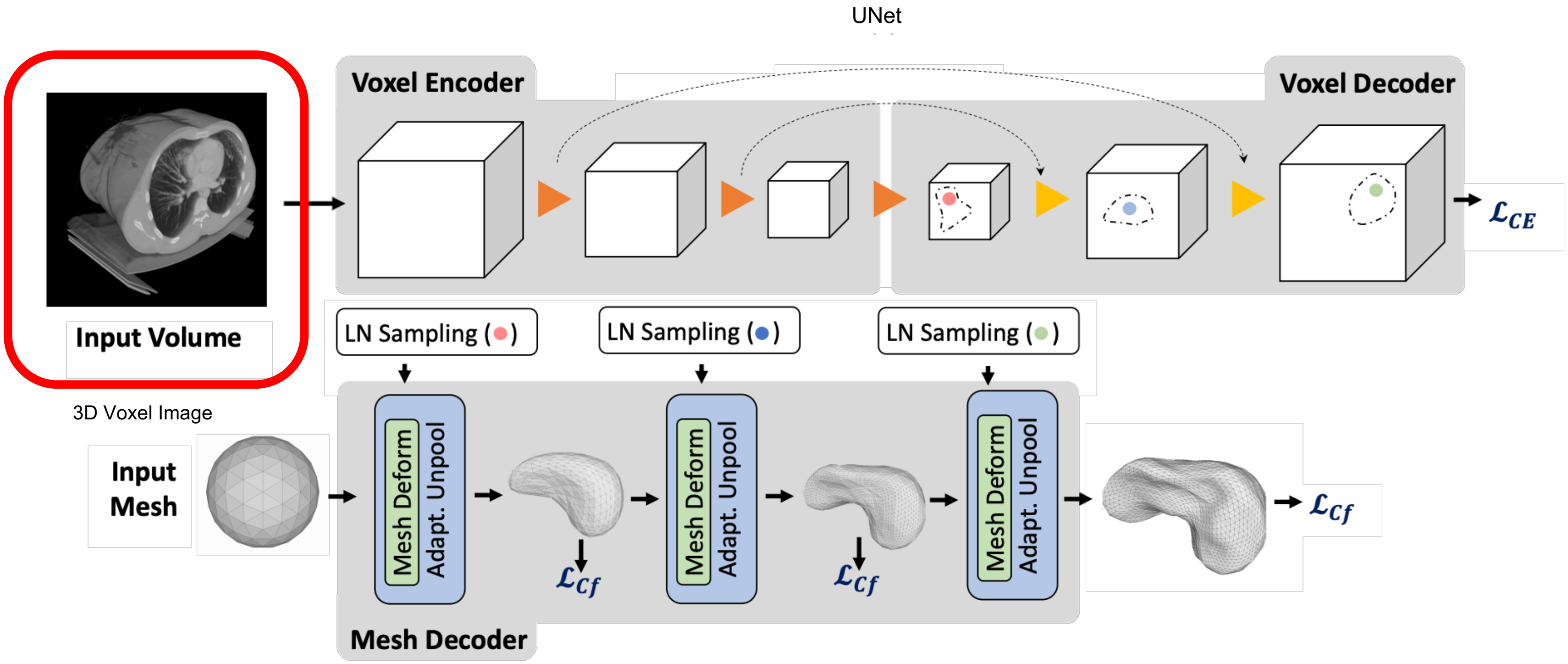


Pixel2Mesh

EXISTING METHODS

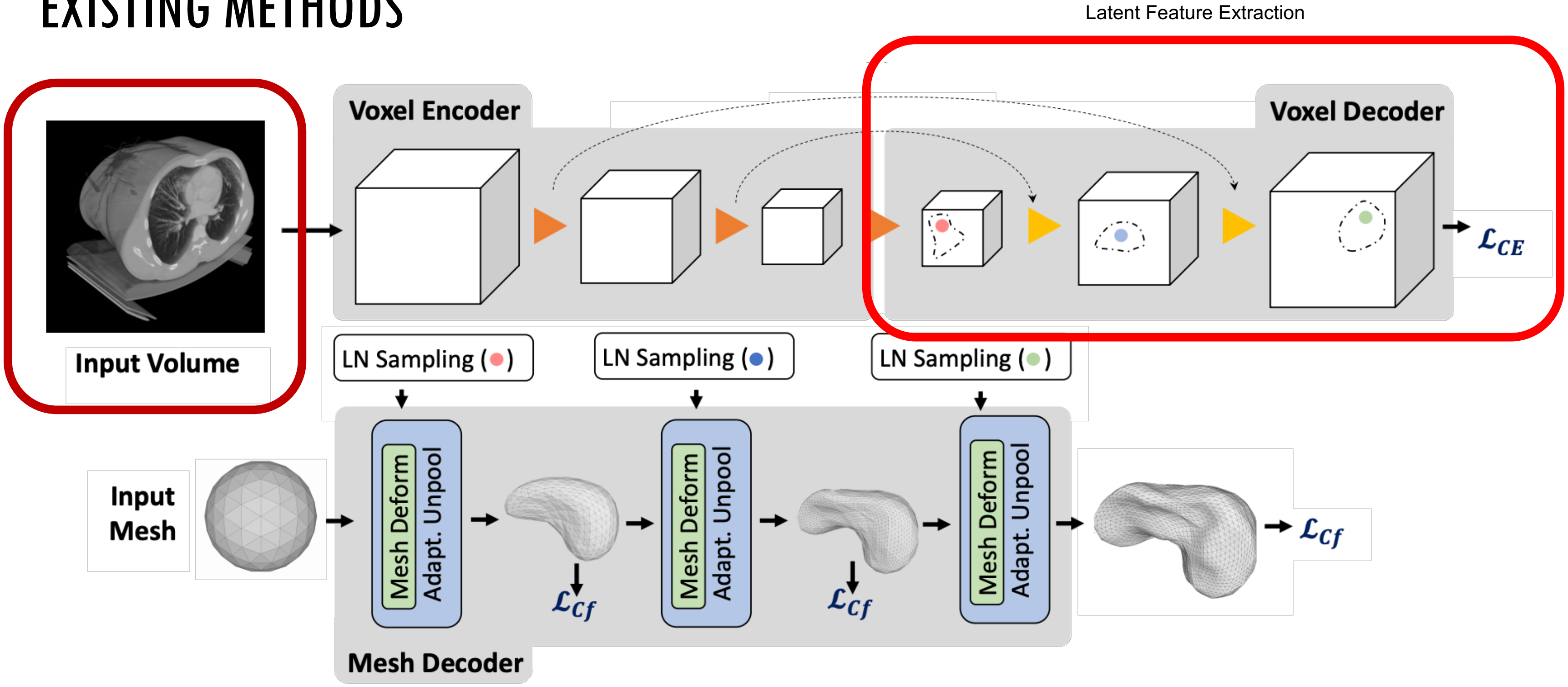
Pixel2Mesh

EXISTING METHODS



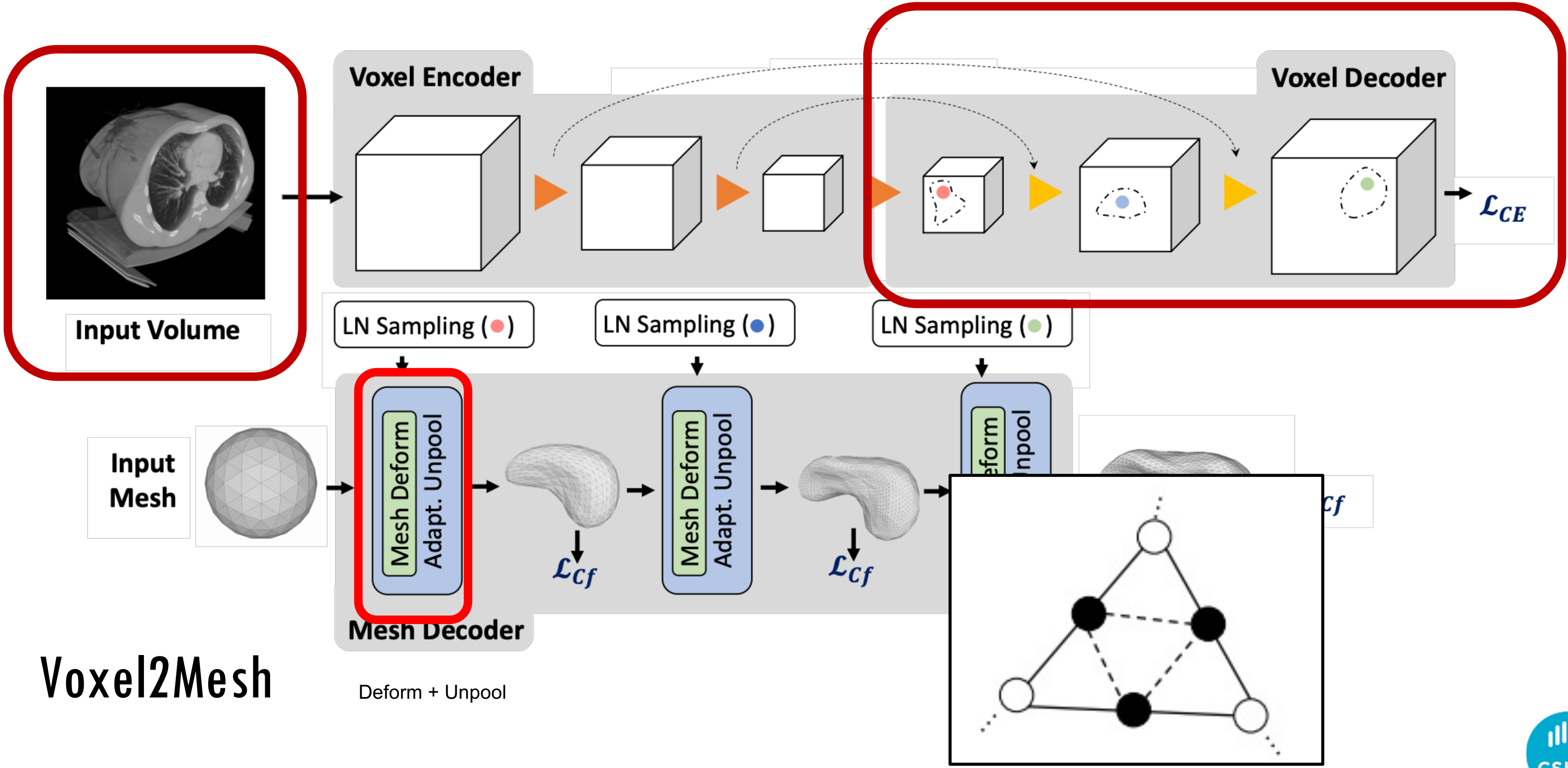
Voxel2Mesh

EXISTING METHODS



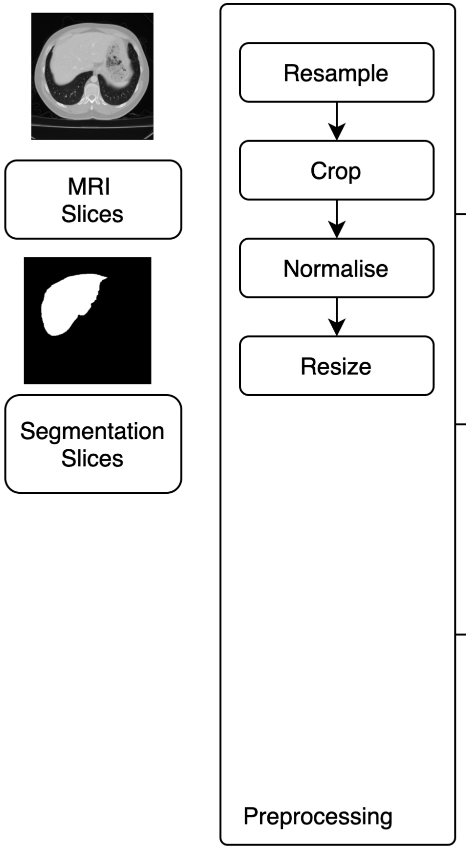
Voxel2Mesh

EXISTING METHODS



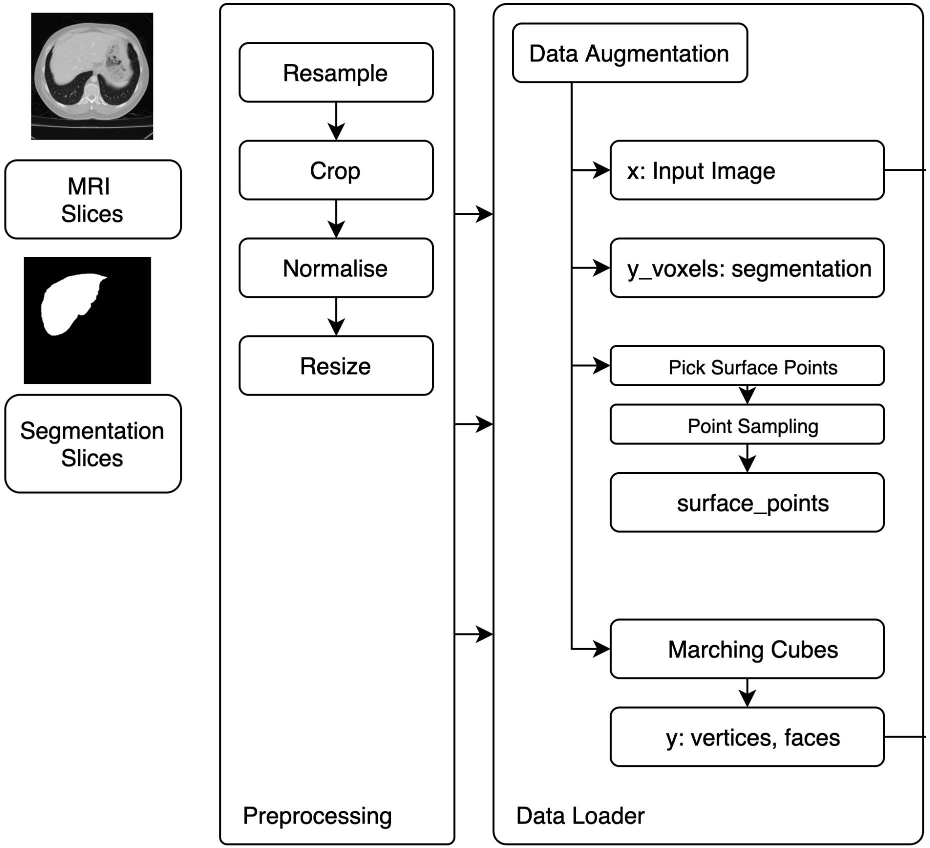
Voxel2Mesh

Deform + Unpool



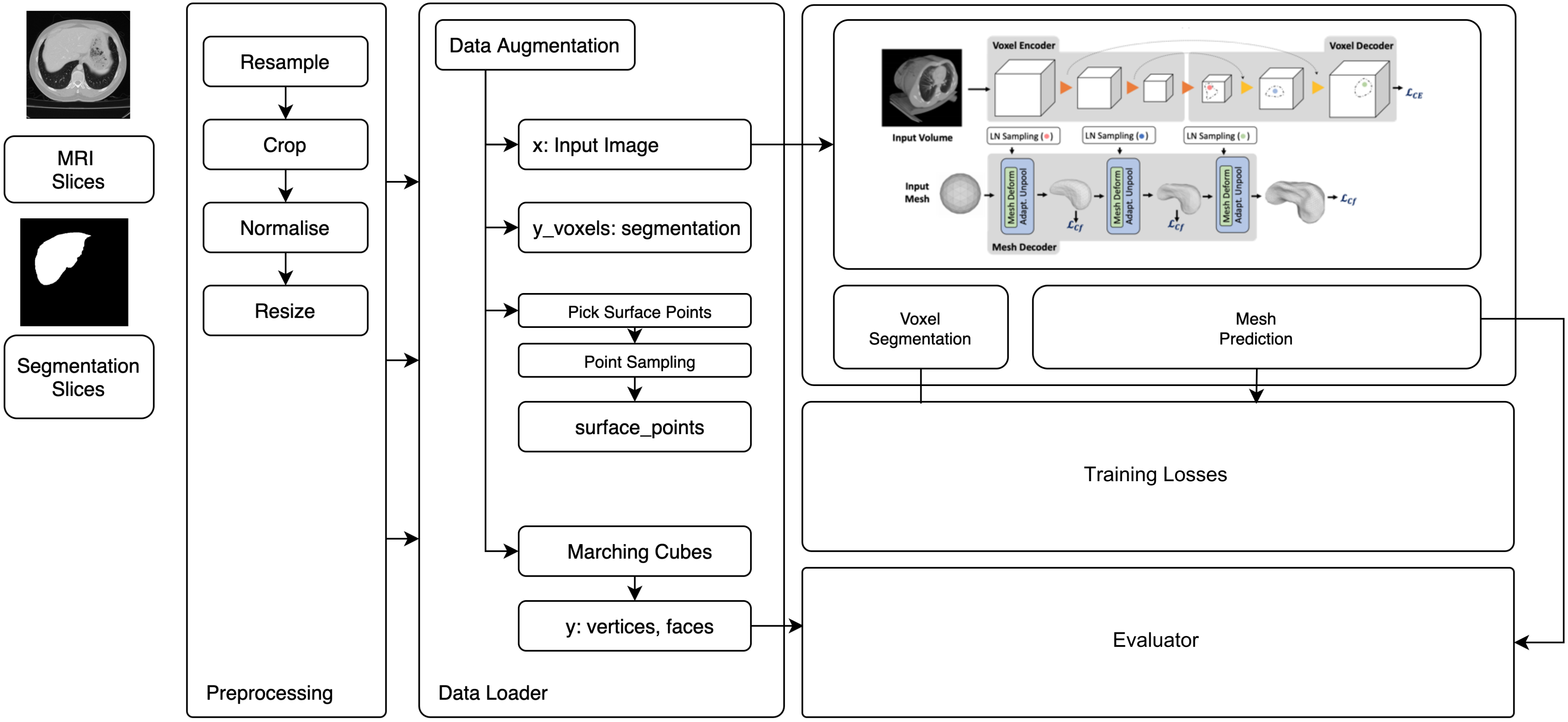
Voxel2Mesh Pipeline

Voxel2Mesh Pipeline



Find segmentation boundaries using dilation algorithm

Generate ground truth mesh for Chamfer loss



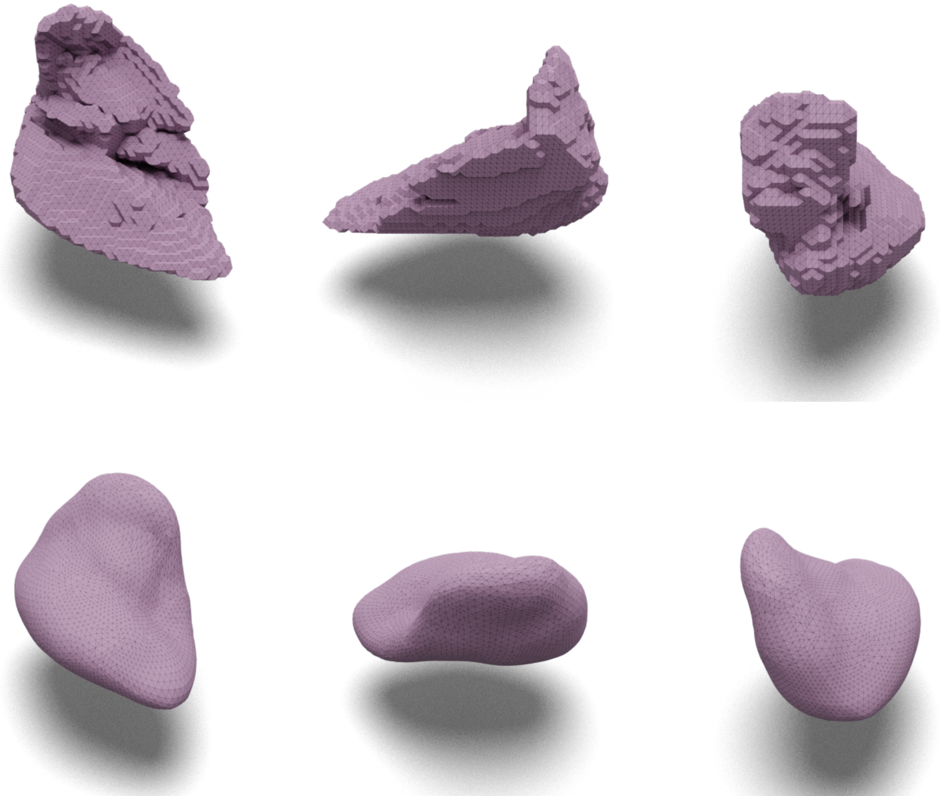
Voxel2Mesh Pipeline

PROJECT AIMS

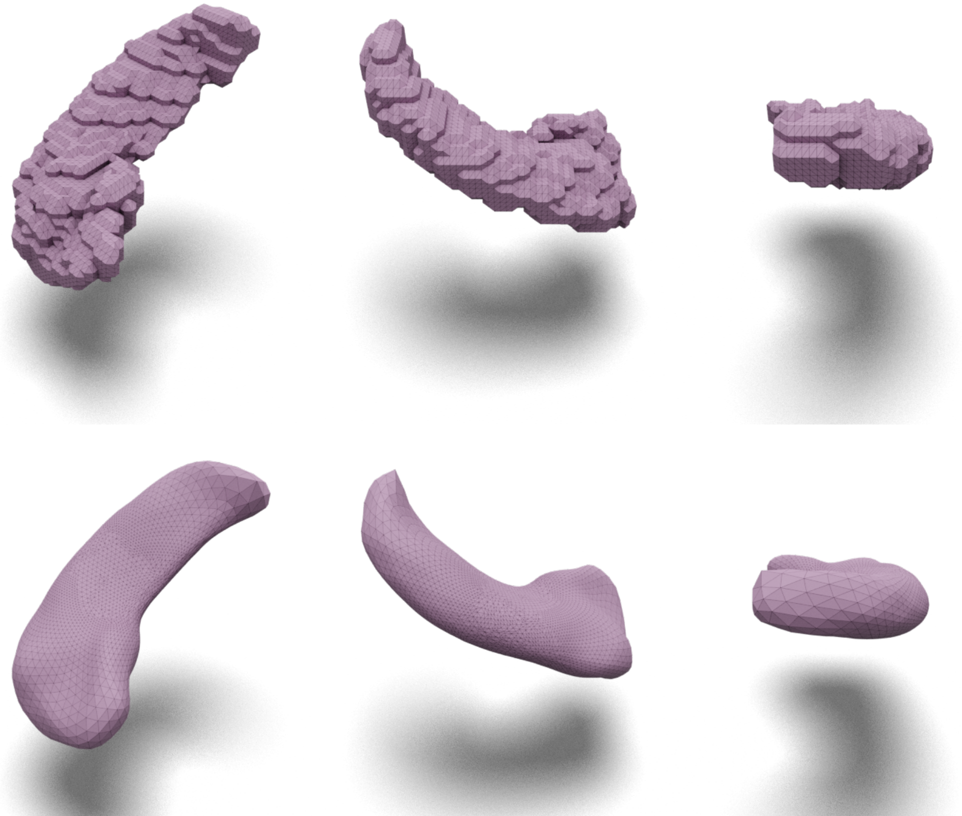
With Voxel2Mesh:

- **Reproduce the results of the original paper**
- Apply Voxel2Mesh to Cortical Surfaces
- (Maybe) see improvements in prediction speed/quality

INITIAL RESULTS - LIVER (CHAOS DATASET)



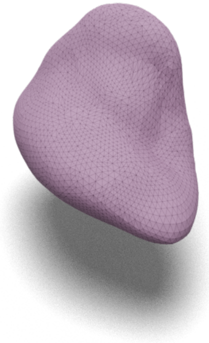
INITIAL RESULTS - HIPPOCAMPUS



INITIAL RESULTS - EVALUATION

- ```
  RuntimeError: CUDA out of memory. Tried to allocate 1.35 GiB.
     (GPU 0; 15.90 GiB total capacity; 13.63 GiB already allocated;
      1.19 GiB free; 13.89 GiB reserved in total by PyTorch)
  ```
  ```

INITIAL RESULTS - EVALUATION

- ```
  RuntimeError: CUDA out of memory. Tried to allocate 1.35 GiB.
     (GPU 0; 15.90 GiB total capacity; 13.63 GiB already allocated;
      1.19 GiB free; 13.89 GiB reserved in total by PyTorch)
  ```

	Liver		Hippocampus	
	IoU	Cf.	IoU	Cf.
PS + UMU	83.3 ± 0.8	3.3×10^{-3}	78.8 ± 1.1	2.9×10^{-3}
HS + UMU	84.2 ± 0.6	2.8×10^{-3}	79.9 ± 0.9	2.3×10^{-3}
LNS + UMU	85.6 ± 0.9	2.1×10^{-3}	81.2 ± 1.2	1.8×10^{-3}
LNS + AMU (Voxel2Mesh)	$\mathbf{86.9 \pm 1.1}$	$\mathbf{1.3 \times 10^{-3}}$	$\mathbf{82.3 \pm 0.9}$	$\mathbf{1.1 \times 10^{-3}}$
Voxel2Mesh on Bracewell	42.5	5.7×10^{-2}	75.4	2.3×10^{-3}

IoU: Rasterized intersection over union e.g. Jaccard
Cf: Chamfer loss or point to point distance
  ```

INITIAL RESULTS - EVALUATION



Main issues:

- Limitations in memory
- Lack of flexibility in model geometry
- Alignment between prediction and original image
- Low down-sampled resolution of images

INITIAL RESULTS - EVALUATION

Learnings from my first machine learning model:

- What a PyTorch machine learning model ‘looks like’
- Scientific code from GitHub…
- Must keep track of trials and experiments
- Anatomical planes and LPS/RAS coordinate systems
- Working with Bracewell

PROJECT AIMS

With Voxel2Mesh:

- Reproduce the results of the original paper
- **Apply Voxel2Mesh to Cortical Surfaces**

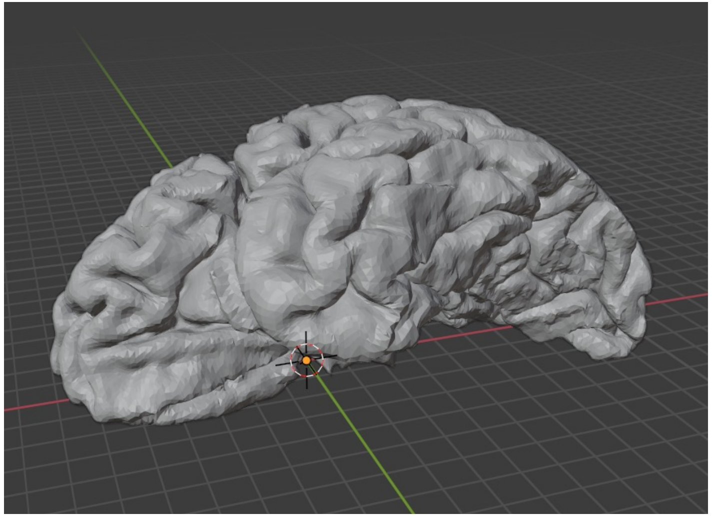
- (Maybe) see improvements in prediction speed/quality

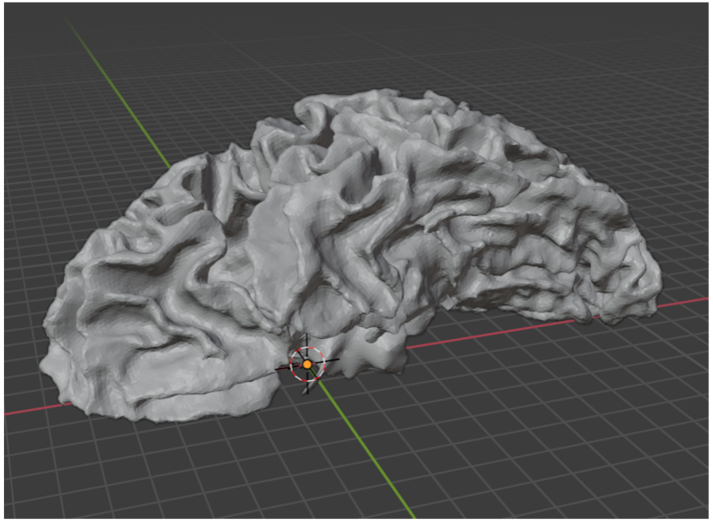
APPLY VOXEL2MESH TO CORTICAL SURFACES

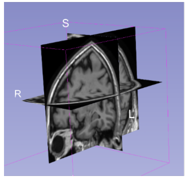
Modifications to Voxel2Mesh:

- Move model to RTX 3090 ☺
- Model a **single hemisphere** of a cortical surface at a time
- Modify ground truth from .PNGs to .OBJ

APPLY VOXEL2MESH TO CORTICAL SURFACES

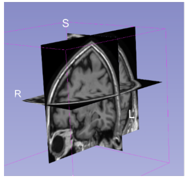






.NII
File

.OBJ
File



.NII
File

.OBJ
File

Resample

↓

Crop/**Align**

↓

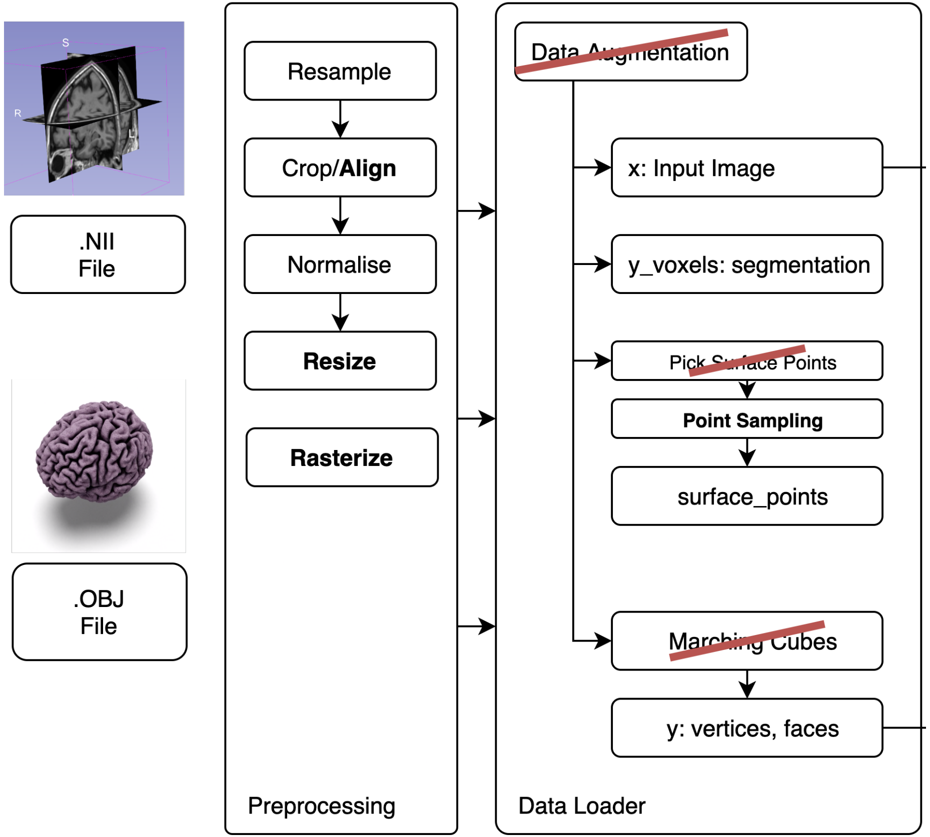
Normalise

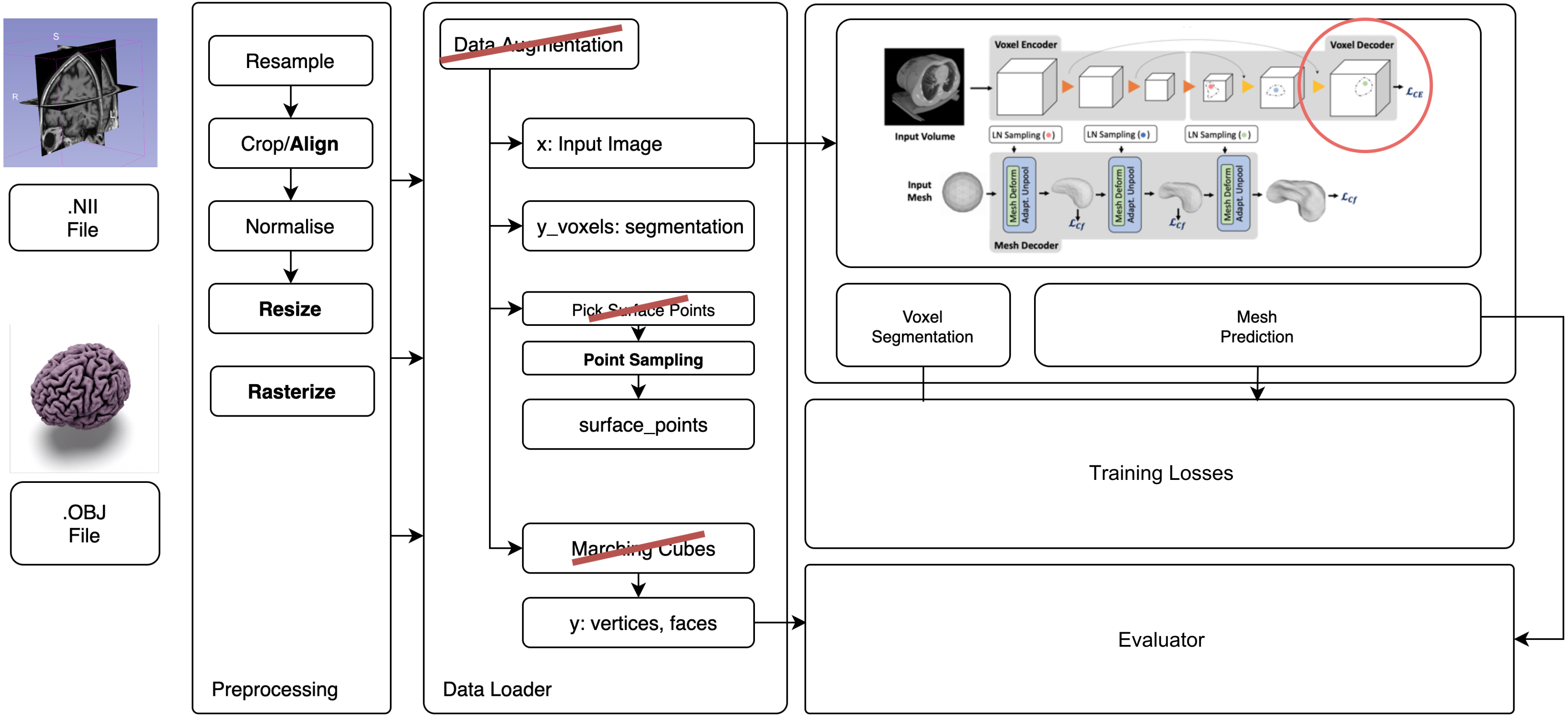
↓

Resize

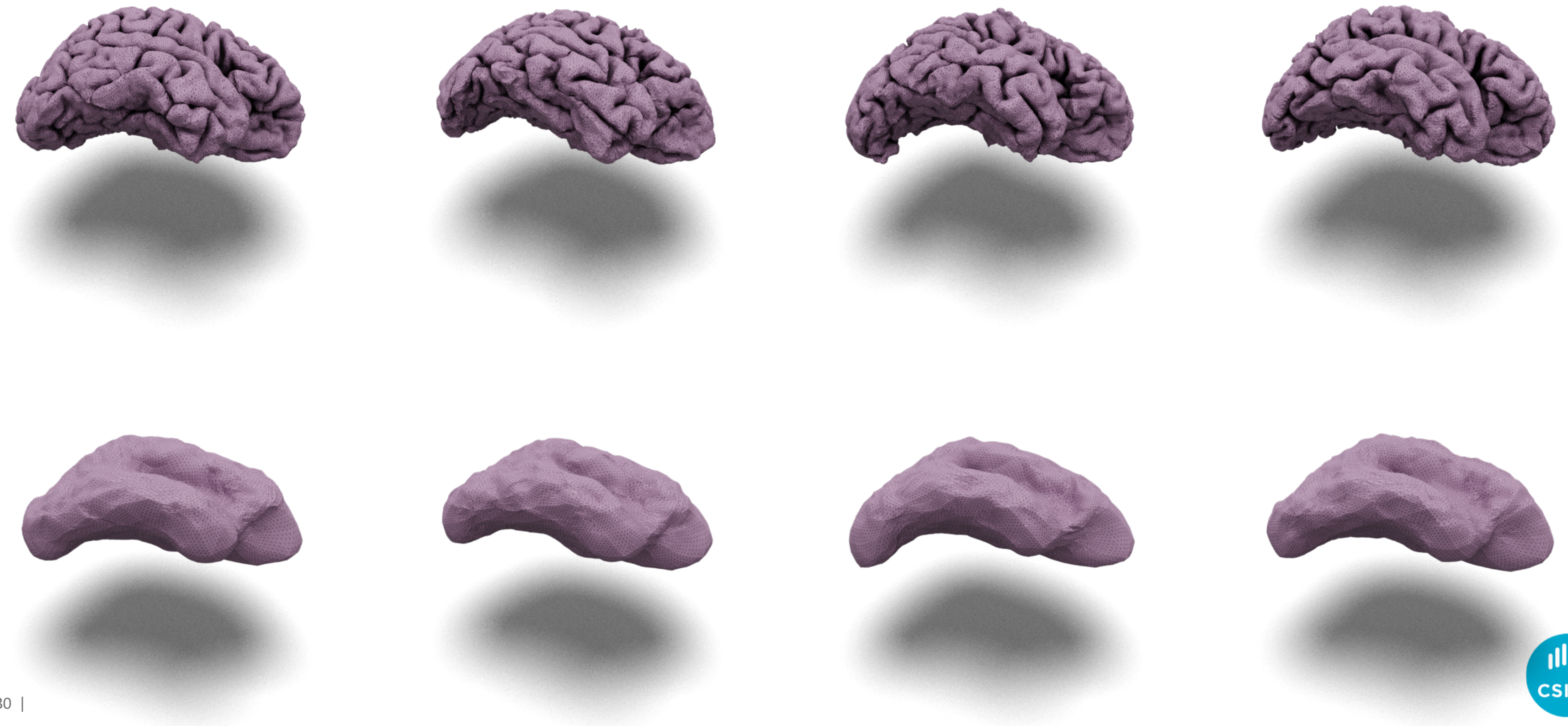
Rasterize

Preprocessing


.NII
File
.OBJ
File
Resample
Crop/**Align**
Normalise
Resize
Rasterize
Preprocessing
~~Data Augmentation~~
x: Input Image
y_voxels: segmentation
~~Pick Surface Points~~
Point Sampling
surface_points
~~Marching Cubes~~
y: vertices, faces
Data Loader

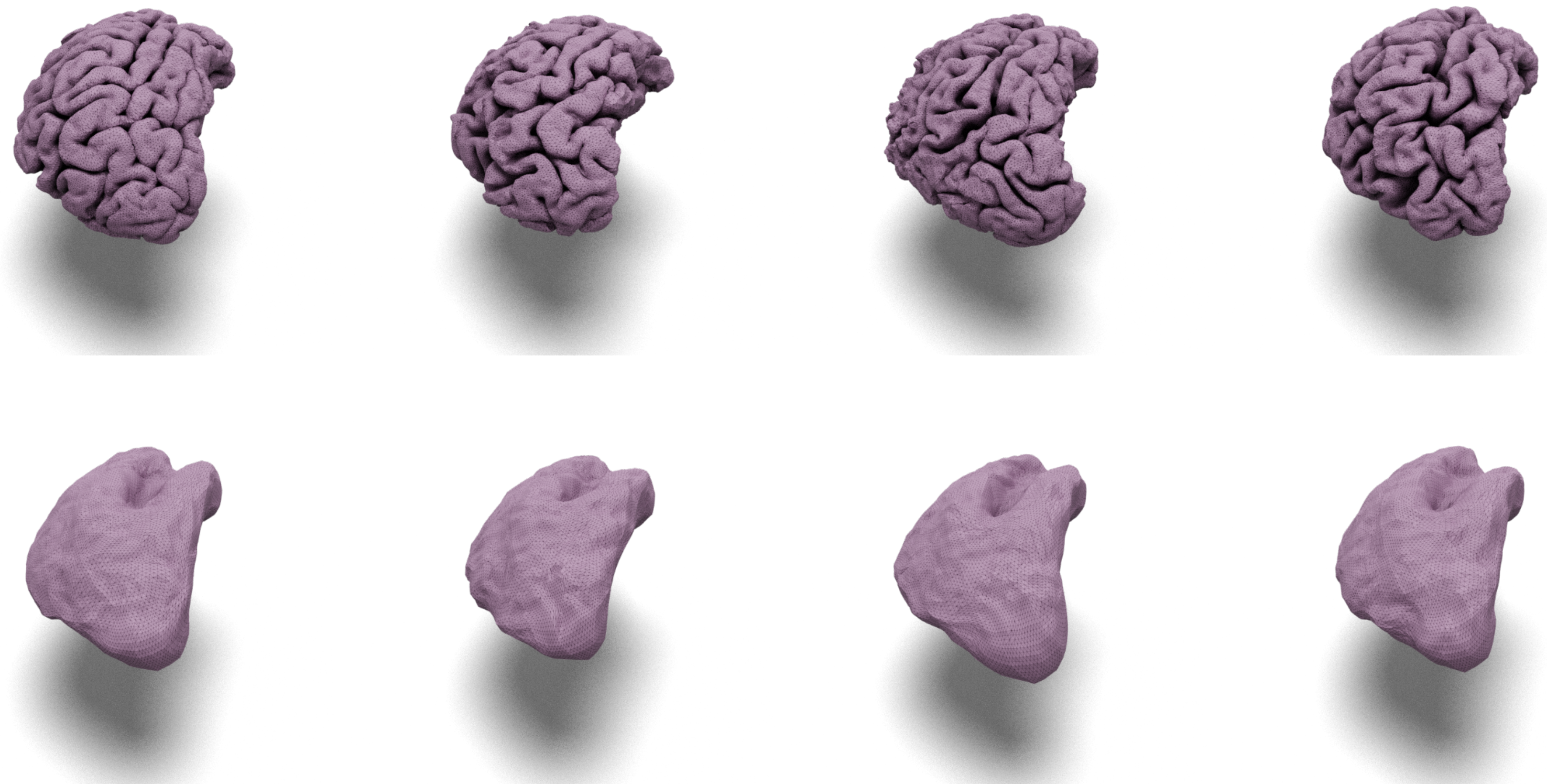

S
R
.NII File
.OBJ File
Resample
Crop/**Align**
Normalise
Resize
Rasterize
Preprocessing
~~Data Augmentation~~
x: Input Image
y_voxels: segmentation
~~Pick Surface Points~~
Point Sampling
surface_points
~~Marching Cubes~~
y: vertices, faces
Data Loader
Voxel Encoder
Voxel Decoder
Input Volume
$\mathcal{L}_{CE}$
LN Sampling
LN Sampling
LN Sampling
Input Mesh
Mesh Deform
Adapt. Unpool
$\mathcal{L}_{cf}$
Mesh Decoder
Voxel Segmentation
Mesh Prediction
Training Losses
Evaluator

APPLY VOXEL2MESH TO CORTICAL SURFACES



APPLY VOXEL2MESH TO CORTICAL SURFACES

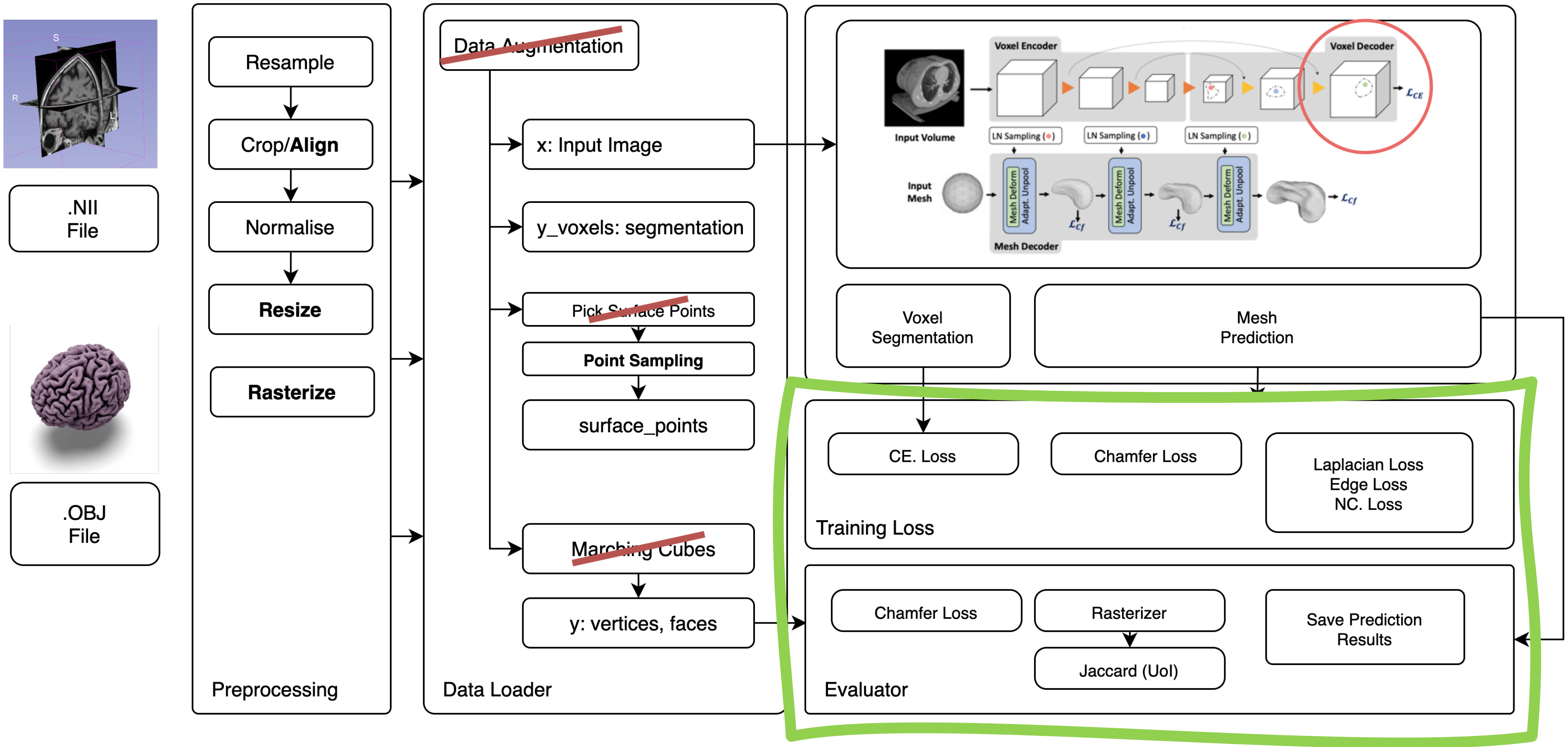
APPLY VOXEL2MESH TO CORTICAL SURFACES



PROJECT AIMS

With Voxel2Mesh:

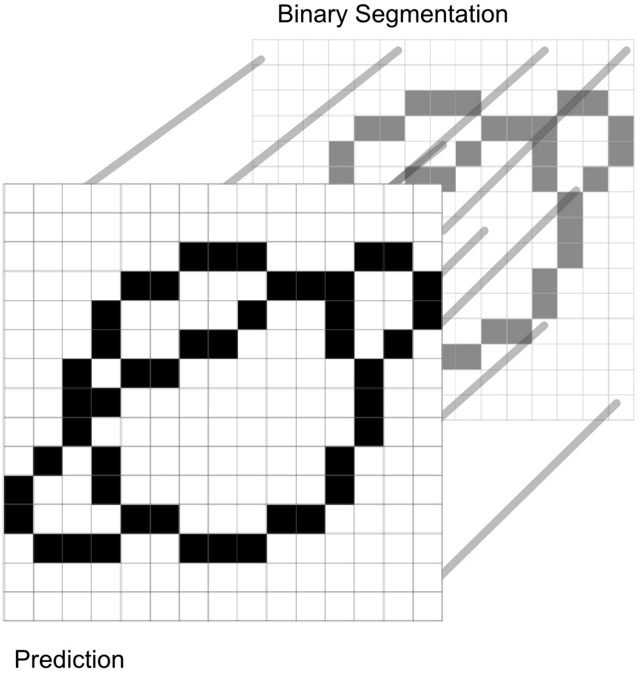
- Reproduce the results of the original paper
- Apply Voxel2Mesh to Cortical Surfaces
- **(Maybe) see improvements in prediction speed/quality**


.NII
File
.OBJ
File
Resample
Crop/Align
Normalise
Resize
Rasterize
Preprocessing
~~Data Augmentation~~
x: Input Image
y_voxels: segmentation
~~Pick Surface Points~~
Point Sampling
surface_points
~~Marching Cubes~~
y: vertices, faces
Data Loader
Voxel Encoder
Voxel Decoder
Input Volume
LN Sampling
Input Mesh
Mesh Deform
Adapt. Unpool
Mesh Decoder
$\mathcal{L}_{CE}$
$\mathcal{L}_{cf}$
Voxel
Segmentation
Mesh
Prediction
CE. Loss
Chamfer Loss
Laplacian Loss
Edge Loss
NC. Loss
Training Loss
Chamfer Loss
Rasterizer
Jaccard (UoI)
Save Prediction
Results
Evaluator

FINE TUNING THE MODEL

$$\text{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right)$$

$$\text{loss} = \frac{\sum_{i=1}^{N} loss(i, class[i])}{\sum_{i=1}^{N} weight[class[i]]}$$



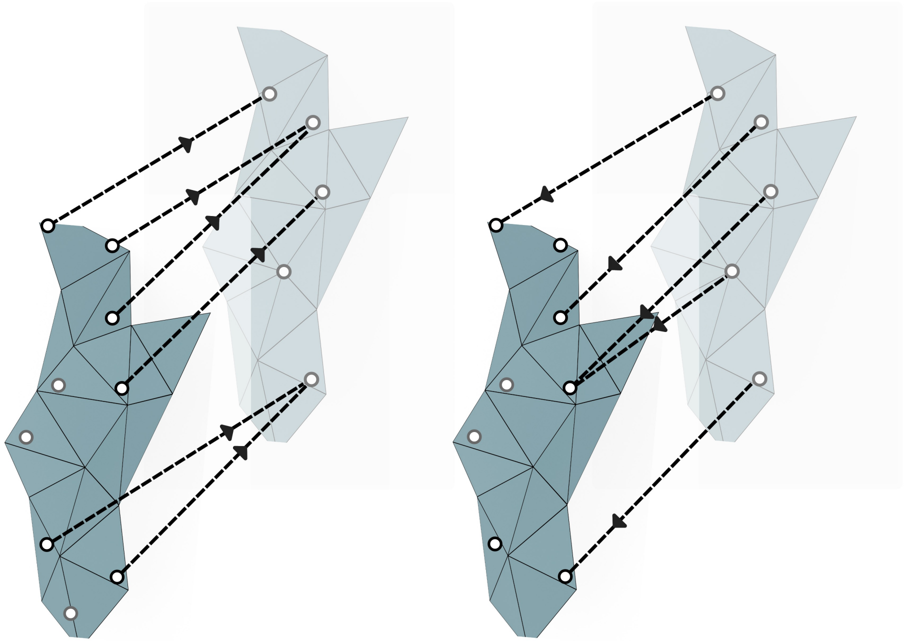
Cross Entropy Loss

FINE TUNING THE MODEL

$$D_{chamfer}(A,B) = \frac{1}{|A|}\sum_{i\in A} d_B(i) + \frac{1}{|B|}\sum_{j\in B} d_A(j)$$

where A, B are sets of points

$d_B(i) =$ Minimum distance between i and some point in B

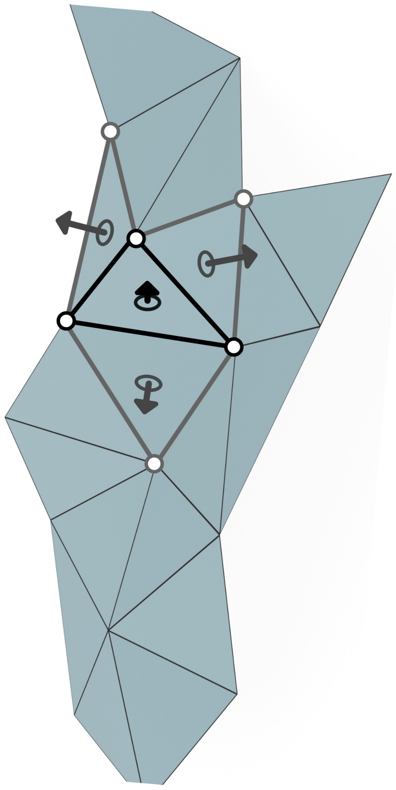


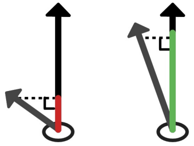
Chamfer Loss

FINE TUNING THE MODEL

$$L_{nc}(F_i, F_j) = 1 - \cos(N_i, N_j)$$

where N_i is the normal of a face F_i





Normal Consistency Loss

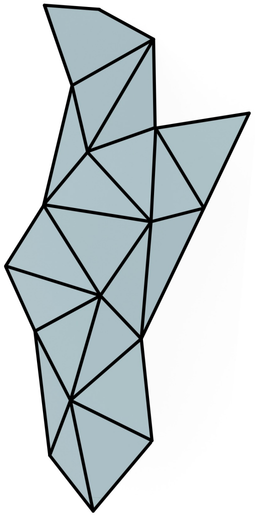
FINE TUNING THE MODEL

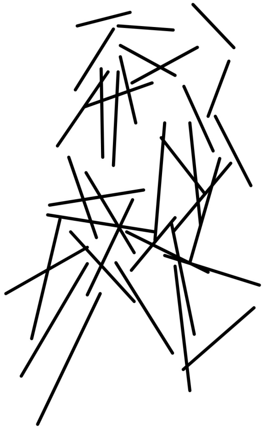
$$\text{Edge Loss} = \sum_i \left| L_{edge}(i) \right|^2$$

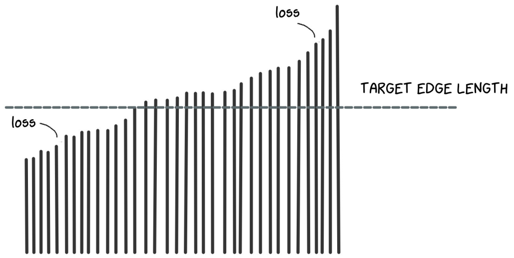
$$Where \;\; L_{edge}(i) = L_{target} - L_i$$

L_{target} is target length

L_i is length of edge i

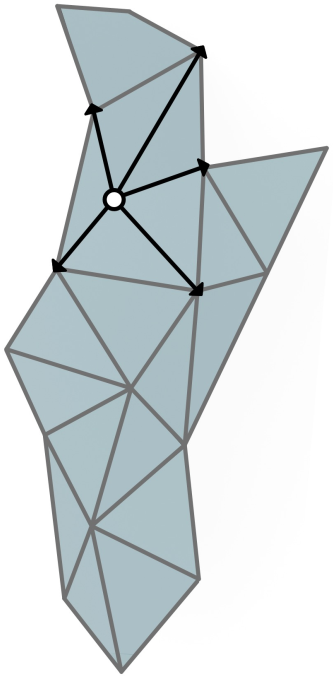


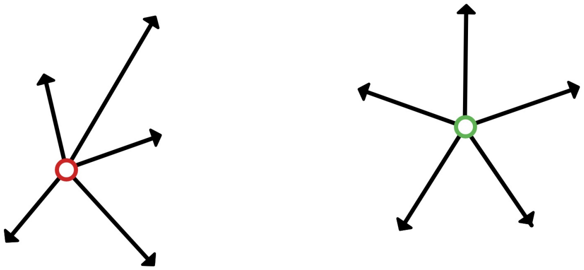




Edge Loss

FINE TUNING THE MODEL





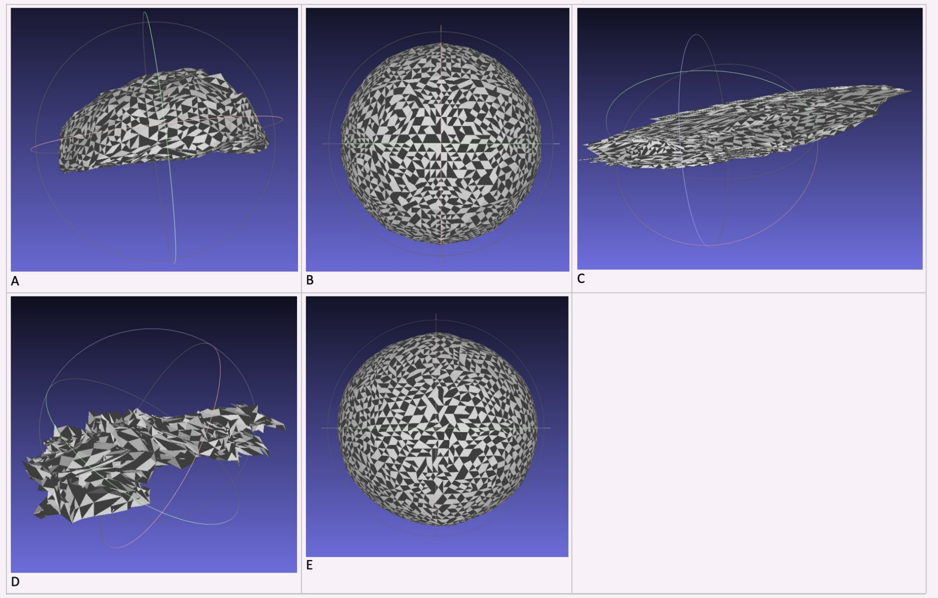
$$L_{lap}(v) = \frac{1}{|\mathcal{N}(v)|} \sum_{w \in \mathcal{N}(v)} w - v$$

where $d(v, w)$ is the distance between two vertices

Laplacian Loss

FINE TUNING THE MODEL

	β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
A	1	0	0	0	0
B	0	1	0	0	0
C	0	0	1	0	0
D	0	0	0	1	0
E	0	0	0	0	1

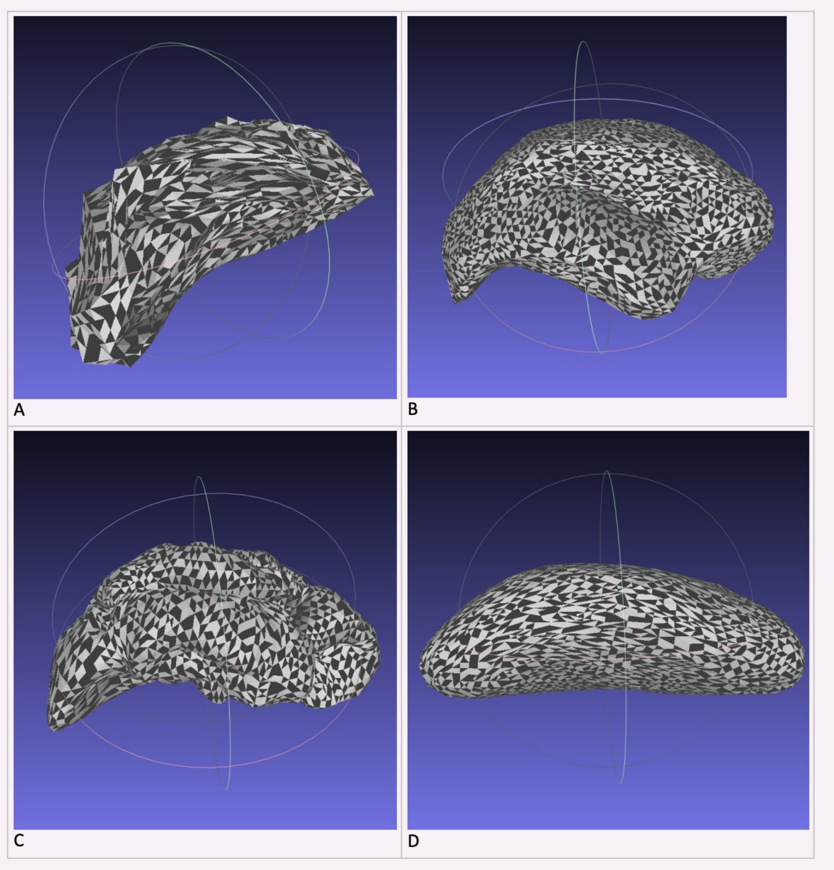


Losses in isolation

FINE TUNING THE MODEL

	β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
A	1	1	0	0	0
B	1	0	1	0	0
C	1	0	0	1	0
D	1	0	0	0	1

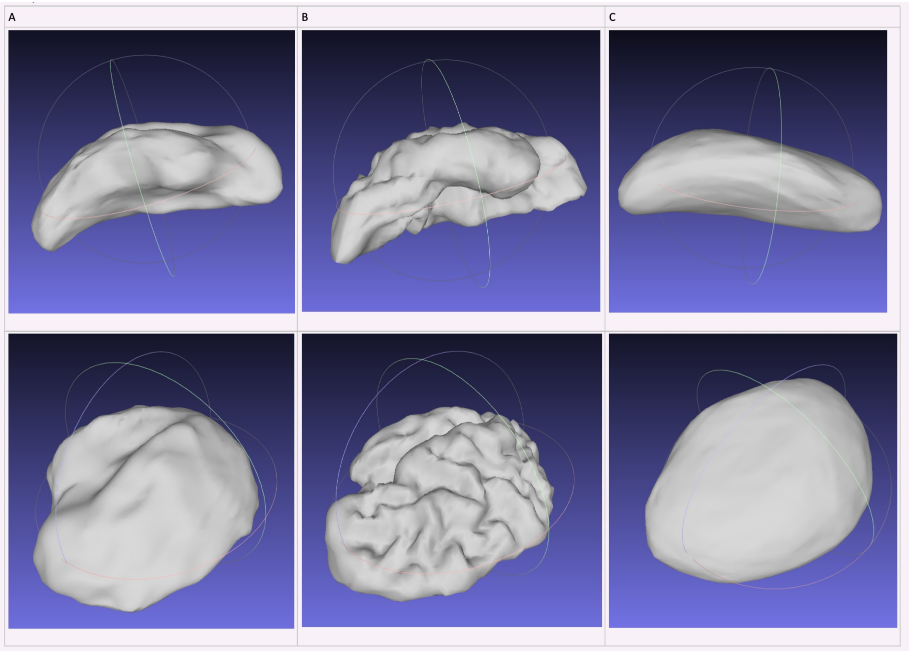
Chamfer loss + isolated losses



FINE TUNING THE MODEL

	β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
A	1	1	1	0	0
B	1	1	0	1	0
C	1	1	0	0	1

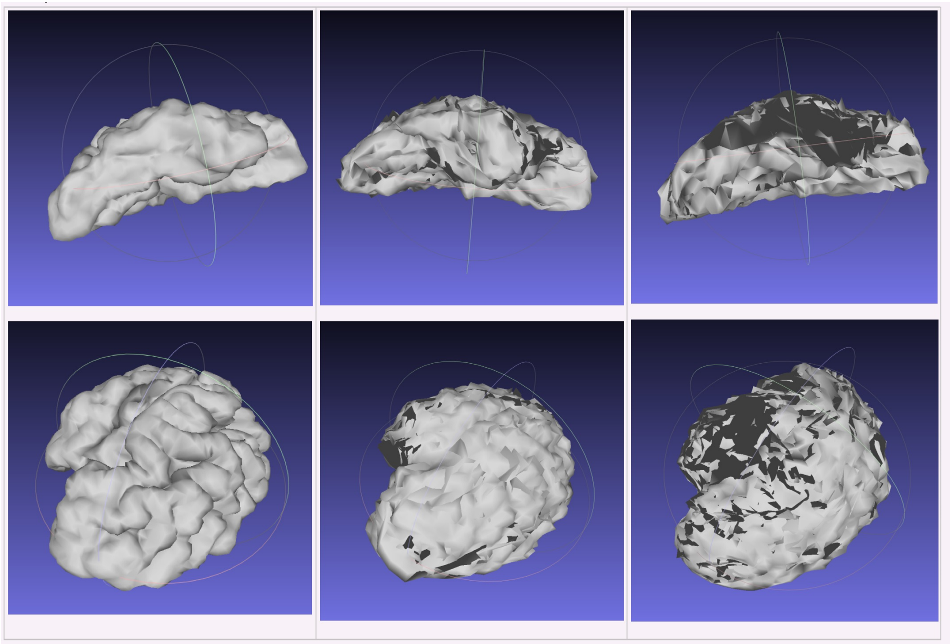
Chamfer Loss and Cross Entropy



FINE TUNING THE MODEL

	β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
A	1	1	0	0.3	0
B	1	1	0	0.03	0
C	1	1	0	0.003	0

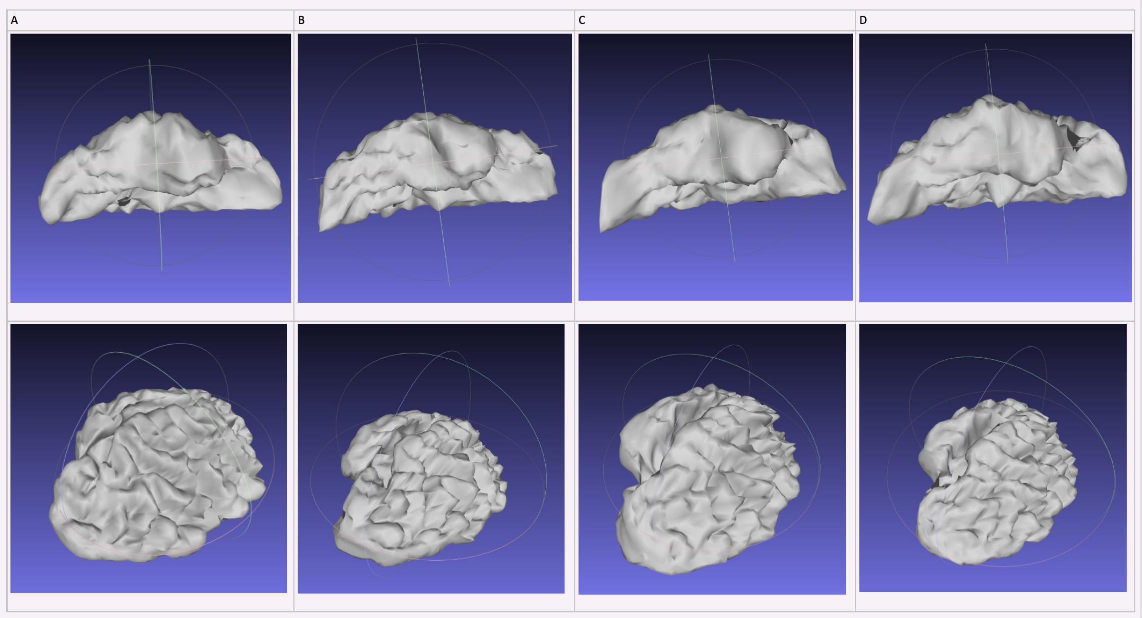
Tune edge loss



FINE TUNING THE MODEL

β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
1	1	0	0.03	0
1	0.5	0	0.03	0
1	0.2	0	0.03	0
1	0.1	0	0.03	0

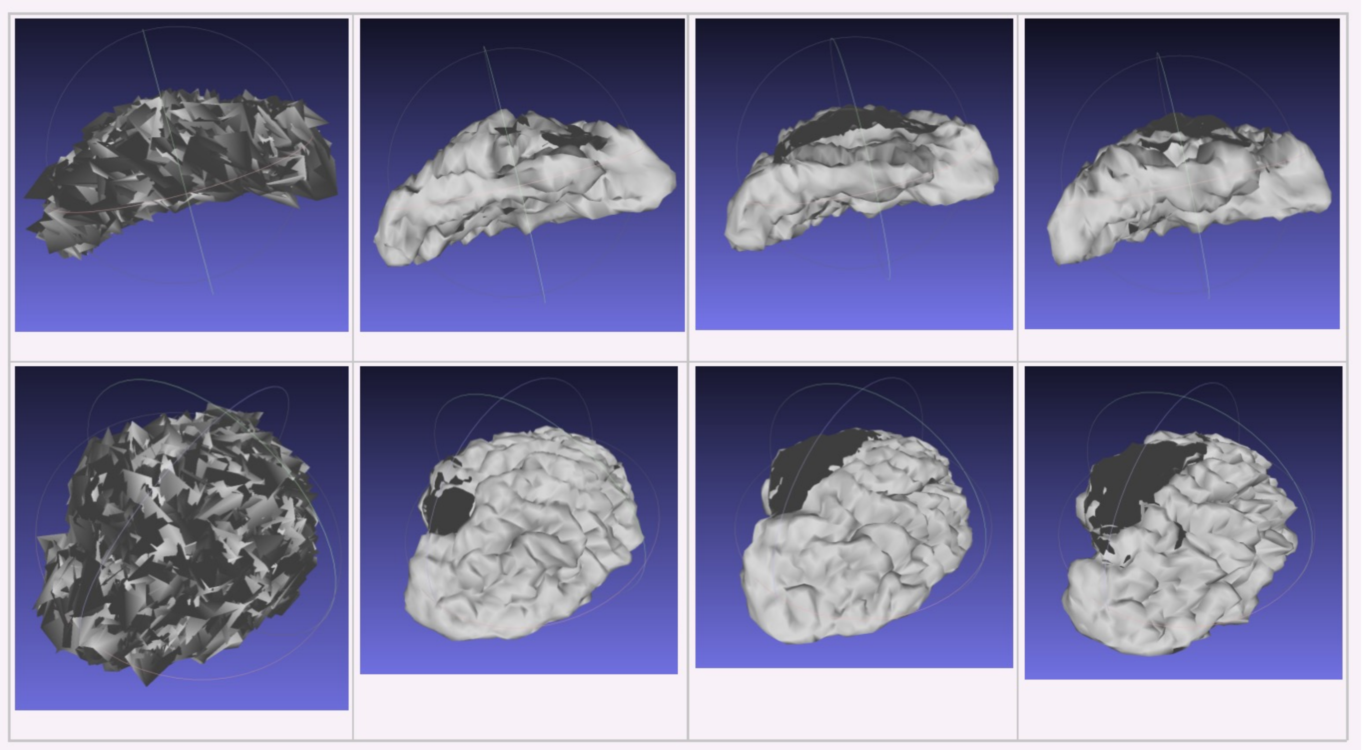
Tune Cross Entropy



FINE TUNING THE MODEL

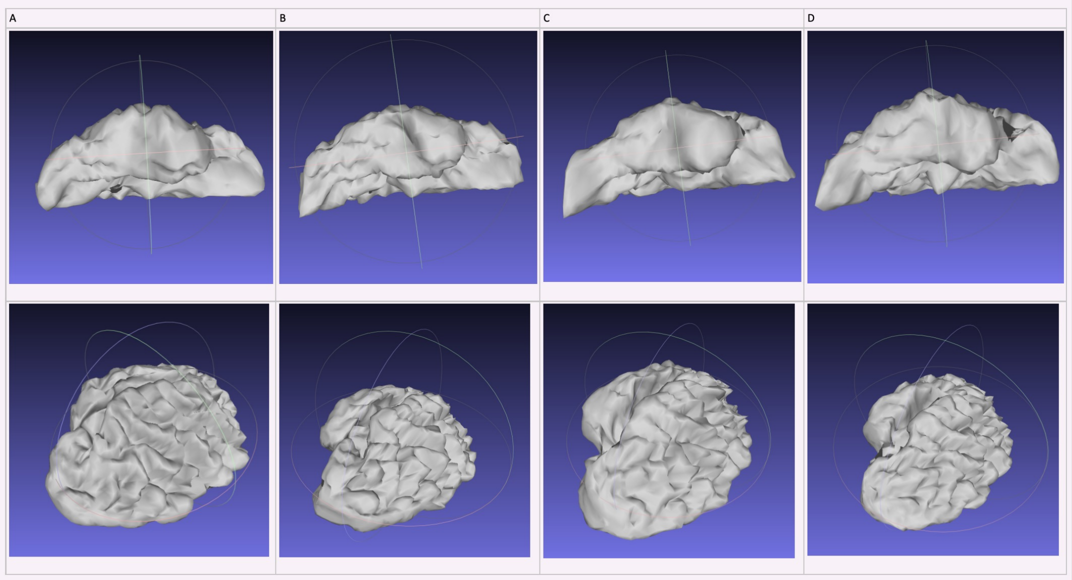
	Edge Length
A	0.3
B	0.03
C	0.02
D	0.04

Tune edge length



FINE TUNING THE MODEL

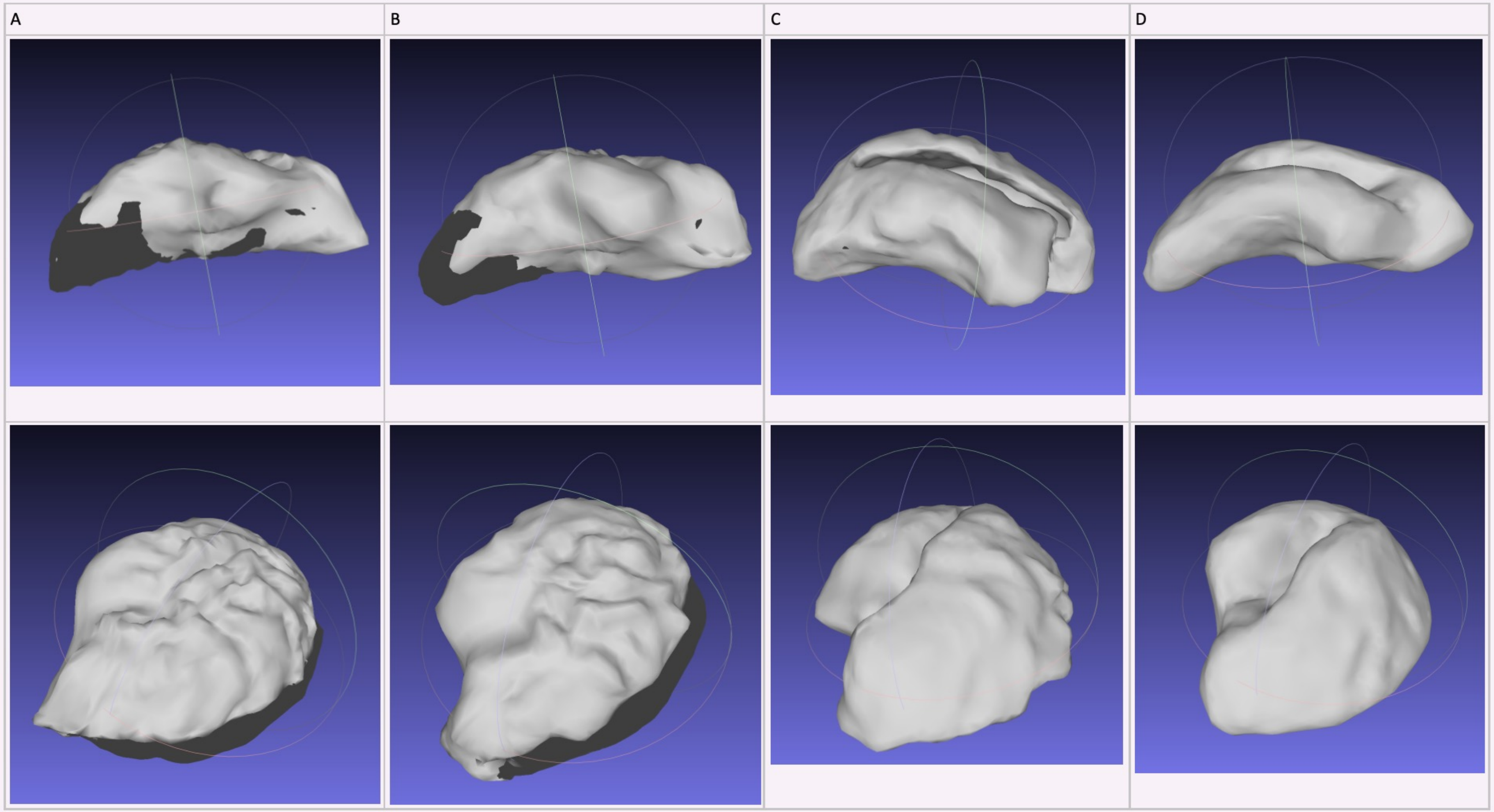
β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
1	1	0	0.3	0.0001
1	1	0	0.3	0.001
1	1	0	0.3	0.01
1	1	0	0.3	0.1
1	1	0.0001	0.3	0
1	1	0.001	0.3	0
1	1	0.01	0.3	0
1	1	0.1	0.3	0
1	1	0	0.3	0
1	1	0	0.2	0
1	1	0	0.1	0



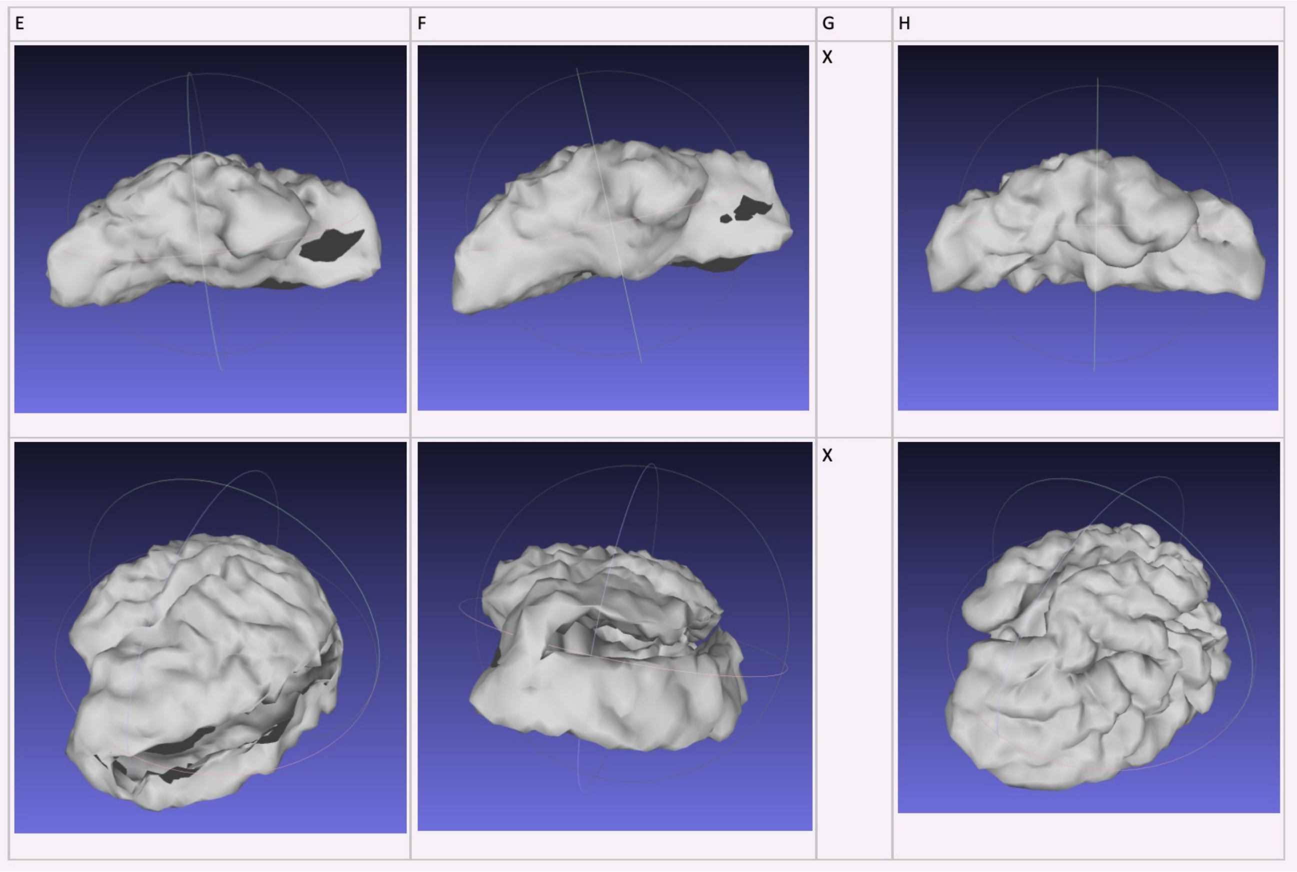
FINE TUNING THE MODEL

β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
1	1	0	0.3	0.0001
1	1	0	0.3	0.001
1	1	0	0.3	0.01
1	1	0	0.3	0.1
1	1	0.0001	0.3	0
1	1	0.001	0.3	0
1	1	0.01	0.3	0
1	1	0.1	0.3	0
1	1	0	0.3	0
1	1	0	0.2	0
1	1	0	0.1	0

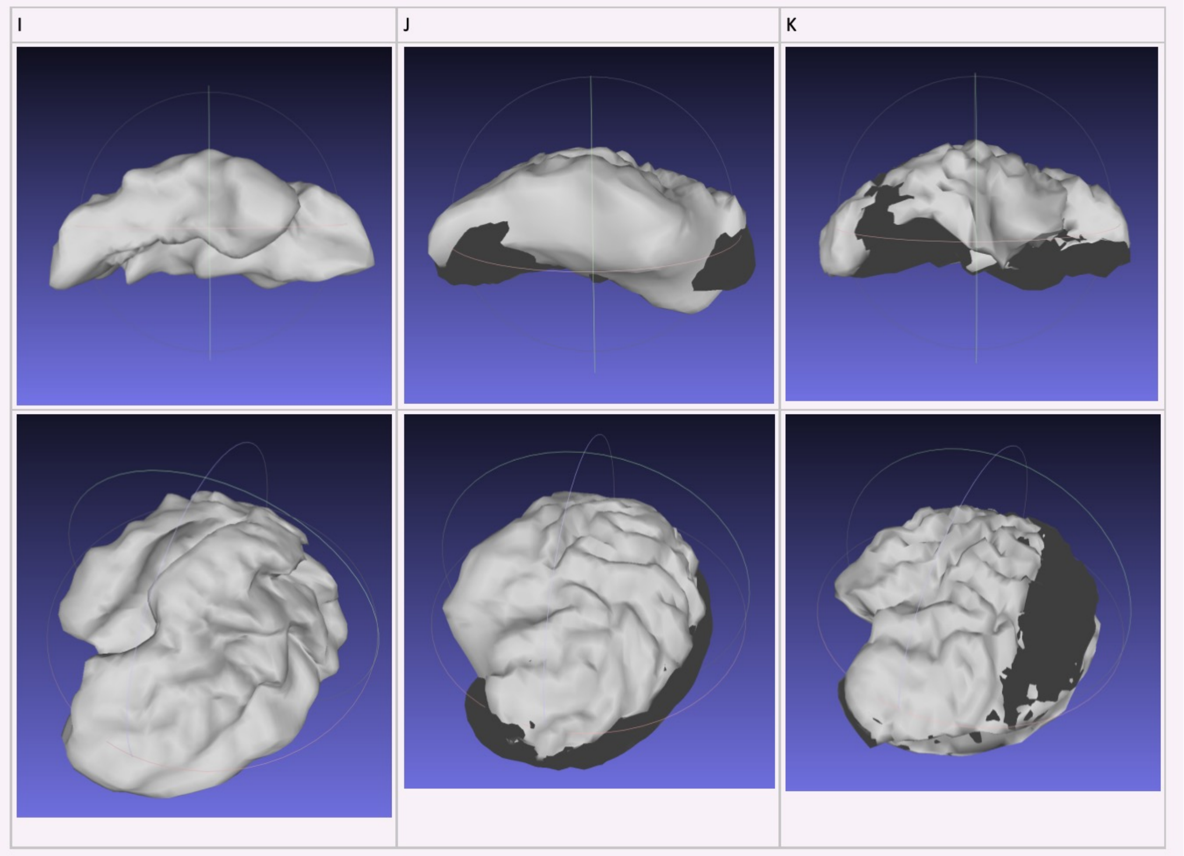
FINE TUNING THE MODEL



FINE TUNING THE MODEL



FINE TUNING THE MODEL



FINE TUNING THE MODEL

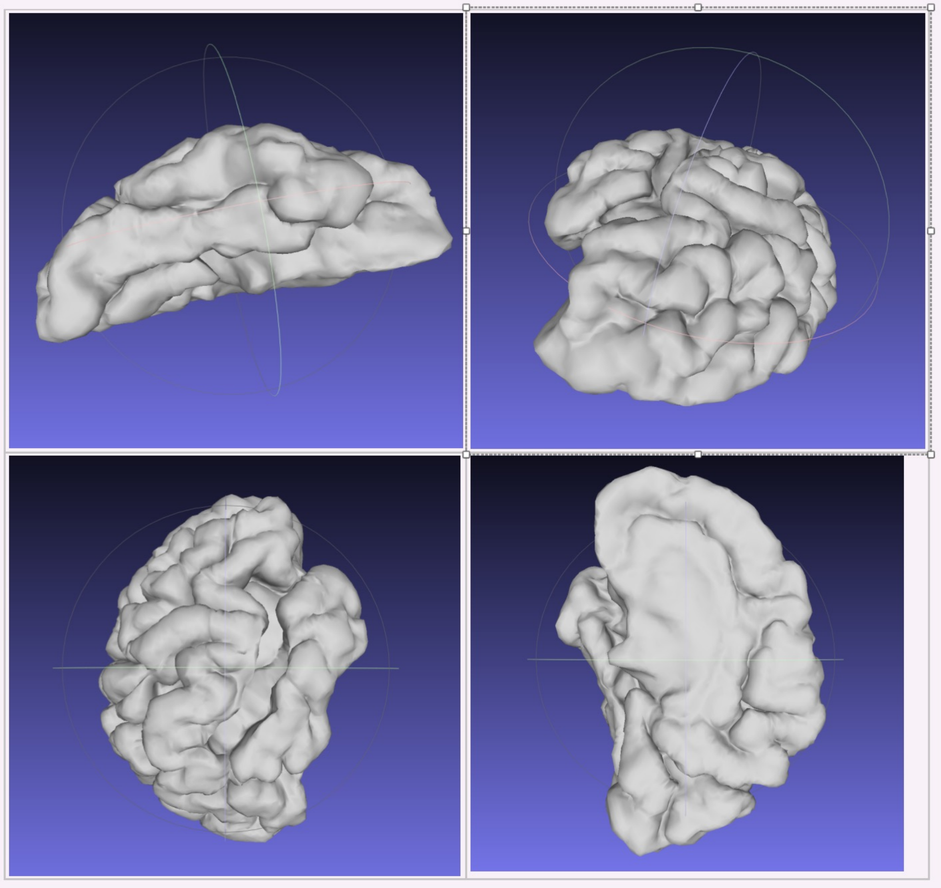
FINE TUNING THE MODEL

β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
1	1	0	0.3	0.0001
1	1	0	0.3	0.001
1	1	0	0.3	0.01
1	1	0	0.3	0.1
1	1	0.0001	0.3	0
1	1	0.001	0.3	0
1	1	0.01	0.3	0
1	1	0.1	0.3	0
1	1	0	0.3	0
1	1	0	0.2	0
1	1	0	0.1	0

FINE TUNING THE MODEL

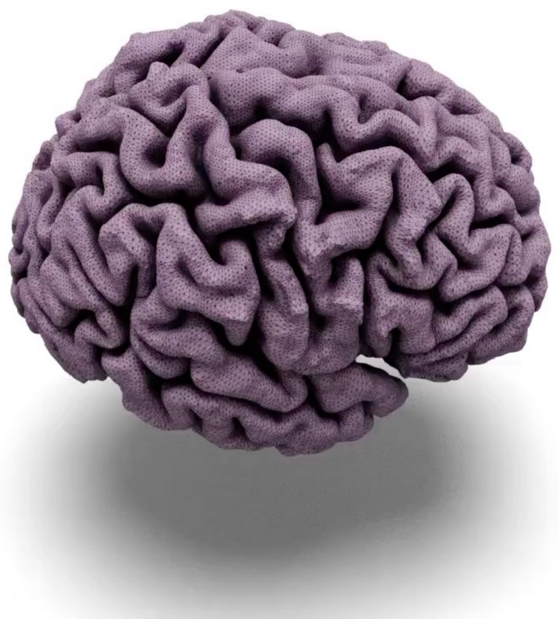
	β_Ch	β_CE	β_Lap	β_Edg	β_Ncn
A	1	1	0	0.3	0

Our best result!



FINAL RESULTS

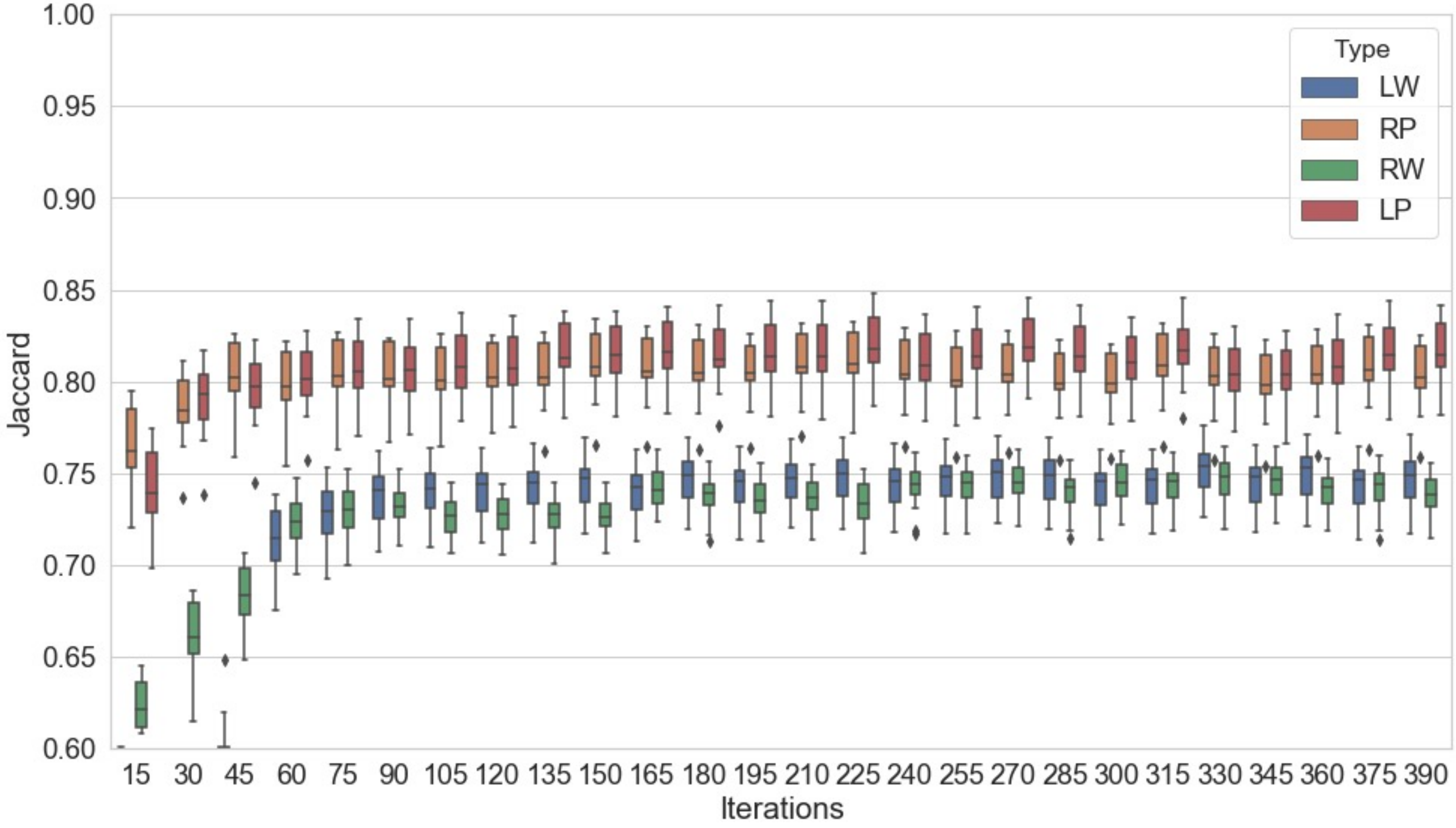
(4 steps on RTX 3090)



FINAL RESULTS

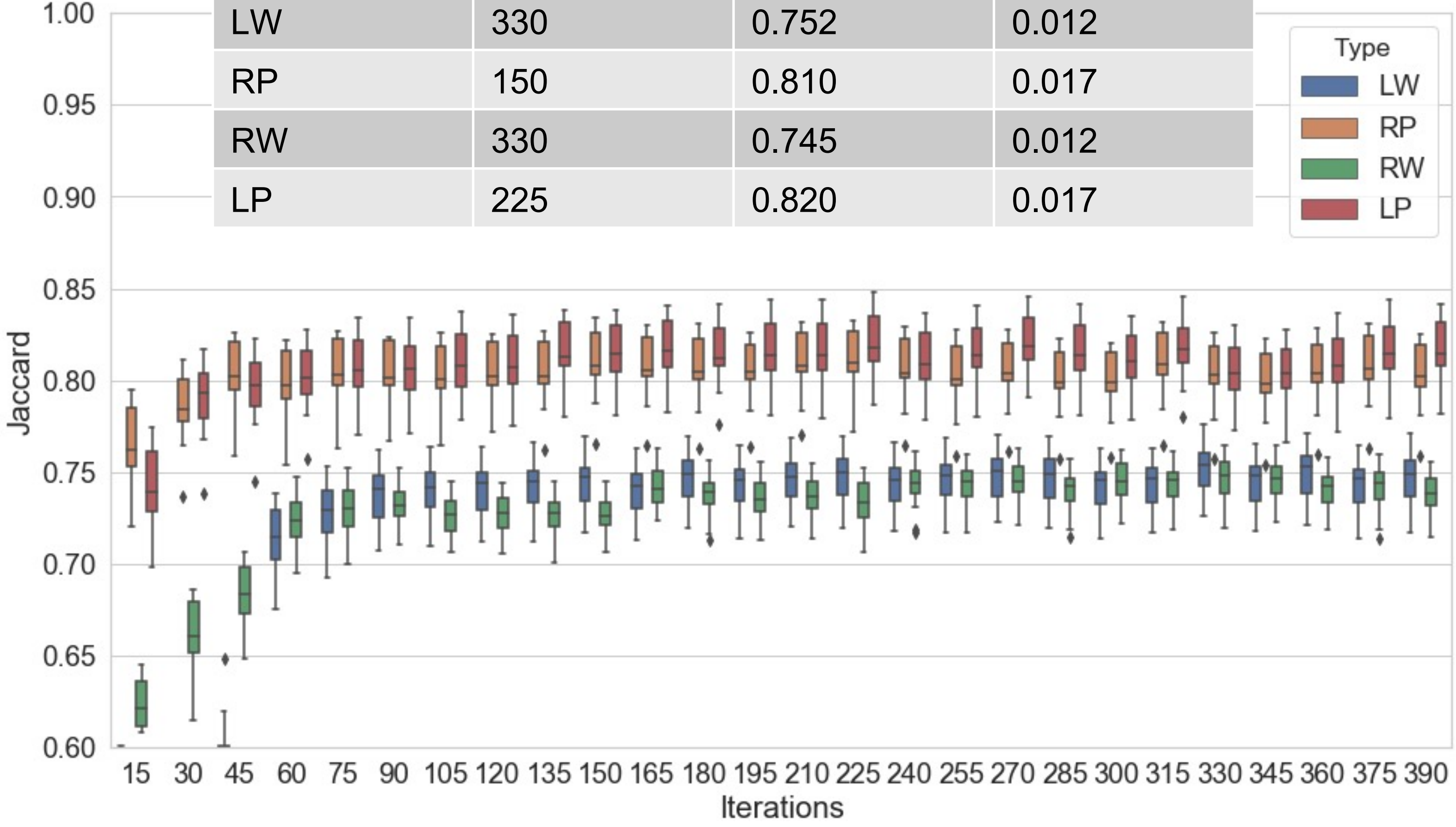
(4 steps on RTX 3090)

NUMERICAL RESULTS

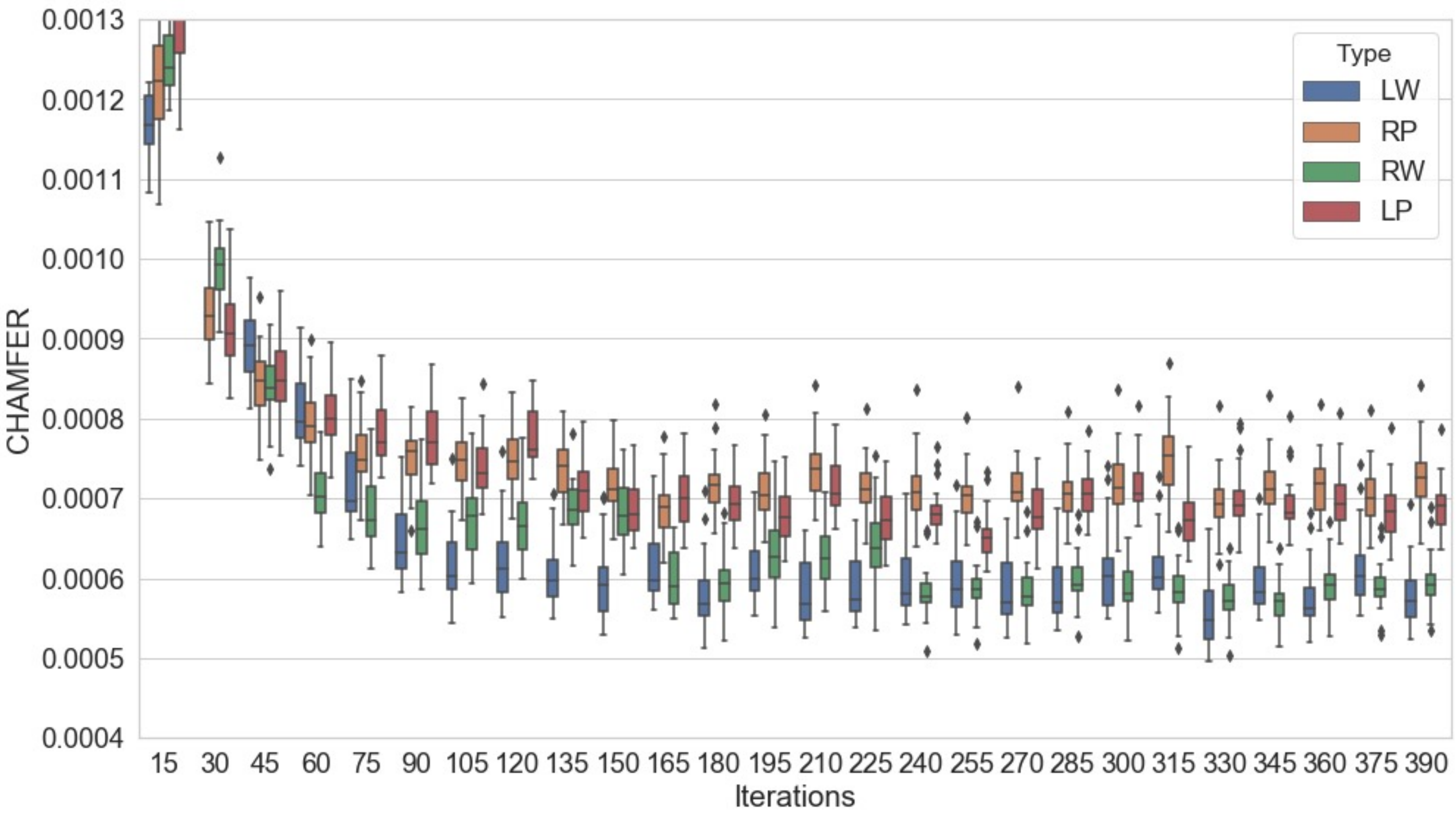


NUMERICAL RESULTS

Type	Top Iteration	Mean	Std.
LW	330	0.752	0.012
RP	150	0.810	0.017
RW	330	0.745	0.012
LP	225	0.820	0.017

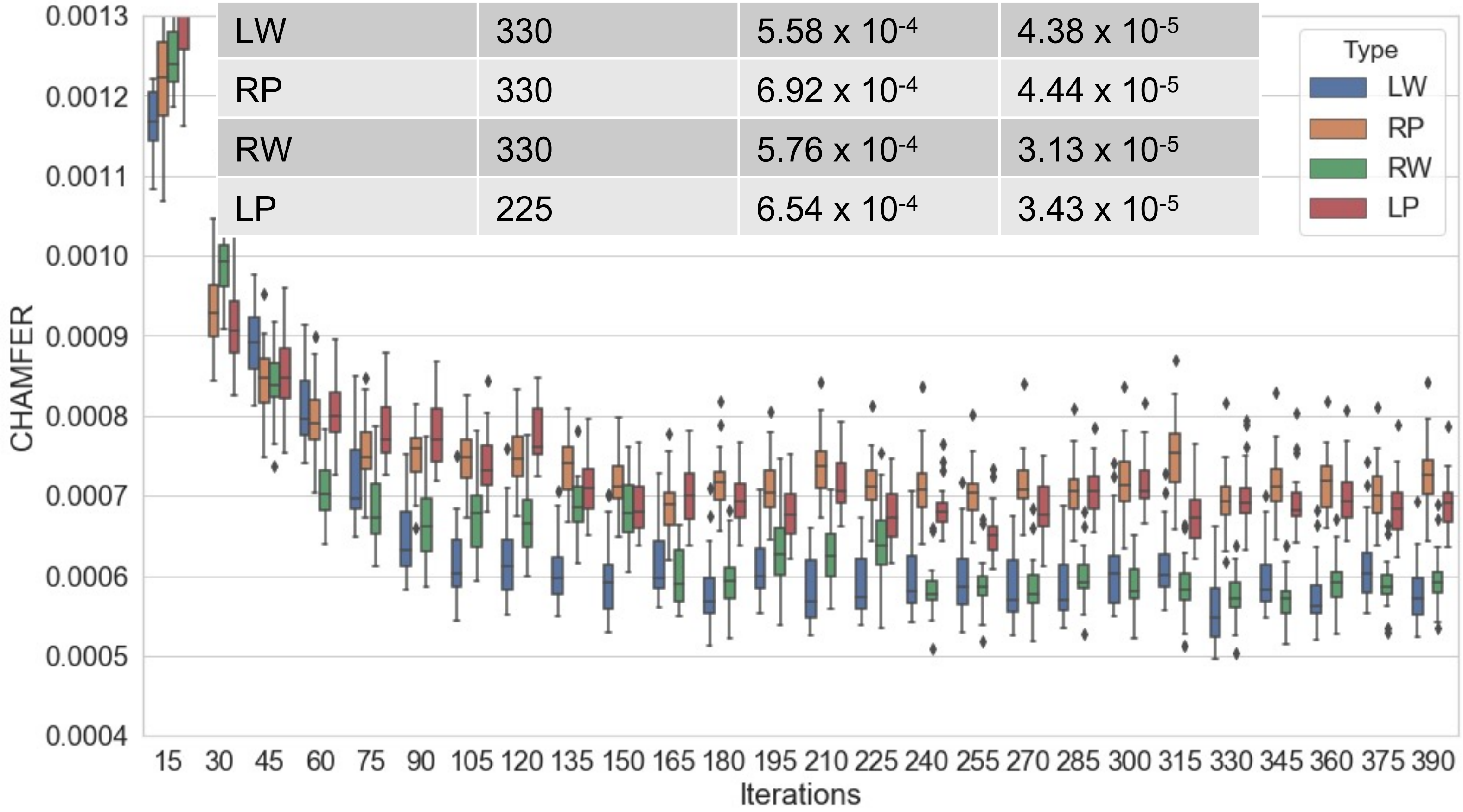


NUMERICAL RESULTS



NUMERICAL RESULTS

Type	Top Iteration	Mean	Std.
LW	330	5.58×10^{-4}	4.38×10^{-5}
RP	330	6.92×10^{-4}	4.44×10^{-5}
RW	330	5.76×10^{-4}	3.13×10^{-5}
LP	225	6.54×10^{-4}	3.43×10^{-5}



NUMERICAL METRICS

- Our results compare favourably to original Voxel2Mesh IoU and Chamfer Loss

	Liver		Hippocampus	
	IoU	Cf.	IoU	Cf.
PS + UMU	83.3 ± 0.8	3.3×10^{-3}	78.8 ± 1.1	2.9×10^{-3}
HS + UMU	84.2 ± 0.6	2.8×10^{-3}	79.9 ± 0.9	2.3×10^{-3}
LNS + UMU	85.6 ± 0.9	2.1×10^{-3}	81.2 ± 1.2	1.8×10^{-3}
LNS + AMU (Voxel2Mesh)	**86.9 ± 1.1**	$\mathbf{1.3 \times 10^{-3}}$	**82.3 ± 0.9**	$\mathbf{1.1 \times 10^{-3}}$

- Significantly decreased inference time **4.8s** in comparison to DeepCSR and FreeSurfer

Method	*Precision on TRT* AD (*mm*)	% > 1 *mm*	% > 2*mm*	*Accuracy on MALC* Dice	VS	*Runtime* (*minutes*)
FreeSurfer	0.241 (±0.291)	2.472	0.983	0.841 (±0.020)	0.953 (±0.027)	373.86 (±47.64)
FastSurfer	0.204 (±0.028)	1.492	0.374	0.834 (±0.021)	0.942 (±0.029)	28.943 (±13.281)
DeepCSR	0.193 (±0.051)	1.266	0.263	0.846 (±0.019)	0.958 (±0.024)	27.824 (±1.393)

NUMERICAL METRICS

- Our results compare favourably to original Voxel2Mesh IoU and Chamfer Loss

	Liver		Hippocampus	
	IoU	Cf.	IoU	Cf.
PS + UMU	83.3 ± 0.8	3.3×10^{-3}	78.8 ± 1.1	2.9×10^{-3}
HS + UMU	84.2 ± 0.6	2.8×10^{-3}	79.9 ± 0.9	2.3×10^{-3}
LNS + UMU	85.6 ± 0.9	2.1×10^{-3}	81.2 ± 1.2	1.8×10^{-3}
LNS + AMU (Voxel2Mesh)	$\mathbf{86.9 \pm 1.1}$	$\mathbf{1.3 \times 10^{-3}}$	$\mathbf{82.3 \pm 0.9}$	$\mathbf{1.1 \times 10^{-3}}$

- Significantly decreased inference time **4.8s** in comparison to DeepCSR and FreeSurfer

	Precision on TRT			*Accuracy on MALC*		*Runtime*
Method	AD (*mm*)	% > 1 *mm*	% > 2*mm*	Dice	VS	(*minutes*)
FreeSurfer	0.241 (±0.291)	2.472	0.983	0.841 (±0.020)	0.953 (±0.027)	373.86 (±47.64)
FastSurfer	0.204 (±0.028)	1.492	0.374	0.834 (±0.021)	0.942 (±0.029)	28.943 (±13.281)
DeepCSR	0.193 (±0.051)	1.266	0.263	0.846 (±0.019)	0.958 (±0.024)	27.824 (±1.393)

NUMERICAL METRICS

- Our results compare favourably to original Voxel2Mesh IoU and Chamfer Loss

	Liver		Hippocampus	
	IoU	Cf.	IoU	Cf.
PS + UMU	83.3 ± 0.8	3.3×10^{-3}	78.8 ± 1.1	2.9×10^{-3}
HS + UMU	84.2 ± 0.6	2.8×10^{-3}	79.9 ± 0.9	2.3×10^{-3}
LNS + UMU	85.6 ± 0.9	2.1×10^{-3}	81.2 ± 1.2	1.8×10^{-3}
LNS + AMU (Voxel2Mesh)	$\mathbf{86.9 \pm 1.1}$	$\mathbf{1.3 \times 10^{-3}}$	$\mathbf{82.3 \pm 0.9}$	$\mathbf{1.1 \times 10^{-3}}$

- Significantly decreased inference time **4.8s** in comparison to DeepCSR and FreeSurfer

Method	*Precision on TRT*			*Accuracy on MALC*		*Runtime*
	AD (*mm*)	% > 1 *mm*	% > 2*mm*	Dice	VS	(*minutes*)
FreeSurfer	0.241 (±0.291)	2.472	0.983	0.841 (±0.020)	0.953 (±0.027)	373.86 (±47.64)
FastSurfer	0.204 (±0.028)	1.492	0.374	0.834 (±0.021)	0.942 (±0.029)	28.943 (±13.281)
DeepCSR	0.193 (±0.051)	1.266	0.263	0.846 (±0.019)	0.958 (±0.024)	27.824 (±1.393)

NUMERICAL METRICS

- Our results compare favourably to original Voxel2Mesh IoU and Chamfer Loss

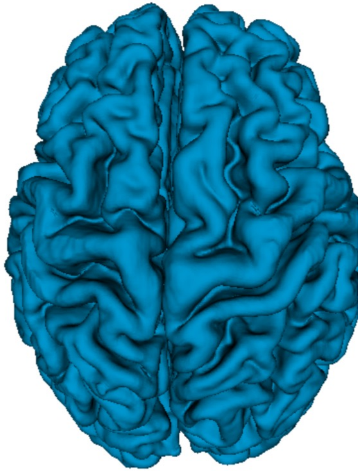
	Liver		Hippocampus	
	IoU	Cf.	IoU	Cf.
PS + UMU	83.3 ± 0.8	3.3×10^{-3}	78.8 ± 1.1	2.9×10^{-3}
HS + UMU	84.2 ± 0.6	2.8×10^{-3}	79.9 ± 0.9	2.3×10^{-3}
LNS + UMU	85.6 ± 0.9	2.1×10^{-3}	81.2 ± 1.2	1.8×10^{-3}
LNS + AMU (Voxel2Mesh)	$\mathbf{86.9 \pm 1.1}$	$\mathbf{1.3 \times 10^{-3}}$	$\mathbf{82.3 \pm 0.9}$	$\mathbf{1.1 \times 10^{-3}}$

- Significantly decreased inference time **4.8s** in comparison to DeepCSR and FreeSurfer

	Precision on TRT			*Accuracy on MALC*		*Runtime*
Method	AD (*mm*)	% > 1 *mm*	% > 2*mm*	Dice	VS	(*minutes*)
FreeSurfer	0.241 (±0.291)	2.472	0.983	0.841 (±0.020)	0.953 (±0.027)	373.86 (±47.64)
FastSurfer	0.204 (±0.028)	1.492	0.374	0.834 (±0.021)	0.942 (±0.029)	28.943 (±13.281)
DeepCSR	0.193 (±0.051)	1.266	0.263	0.846 (±0.019)	0.958 (±0.024)	27.824 (±1.393)

COMPARISON TO DEEPCSR

- 14.7K vs. 900K vertices
- 29.4K vs. 1.9M faces
- Quality/detail is no comparison



LEARNINGS AND FURTHER IMPROVEMENTS

- **Mesh deformation** is able to successfully generate models of cortical surfaces
- Voxel2Mesh is fast
- The biggest limitation to Voxel2Mesh is GPU memory usage
- CorticalFlow

FINAL RESULTS

CSIRO

Australia's National Science Agency

Cortical Surface Retrieval via Deformable Models

A Vacation Student Project

Darren Fu, Rodrigo Santa Cruz, Leo Lebrat, Jurgen Fripp, and Olivier Salvado

d.fu@uq.net.au

THE AUSTRALIAN E-HEALTH RESEARCH CENTRE

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

QUT Queensland University of Technology