Brain Segmentation

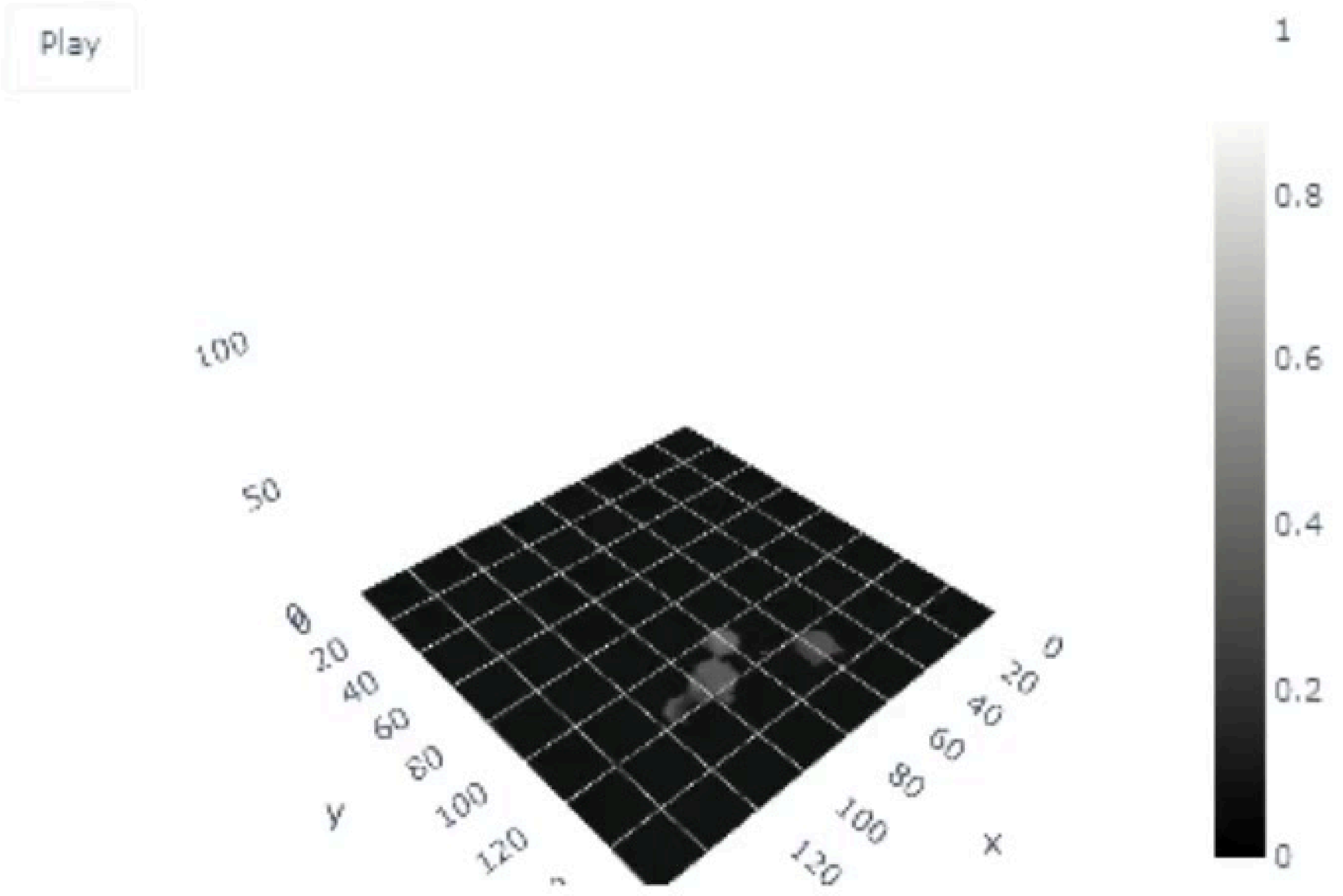
Deep Learning project

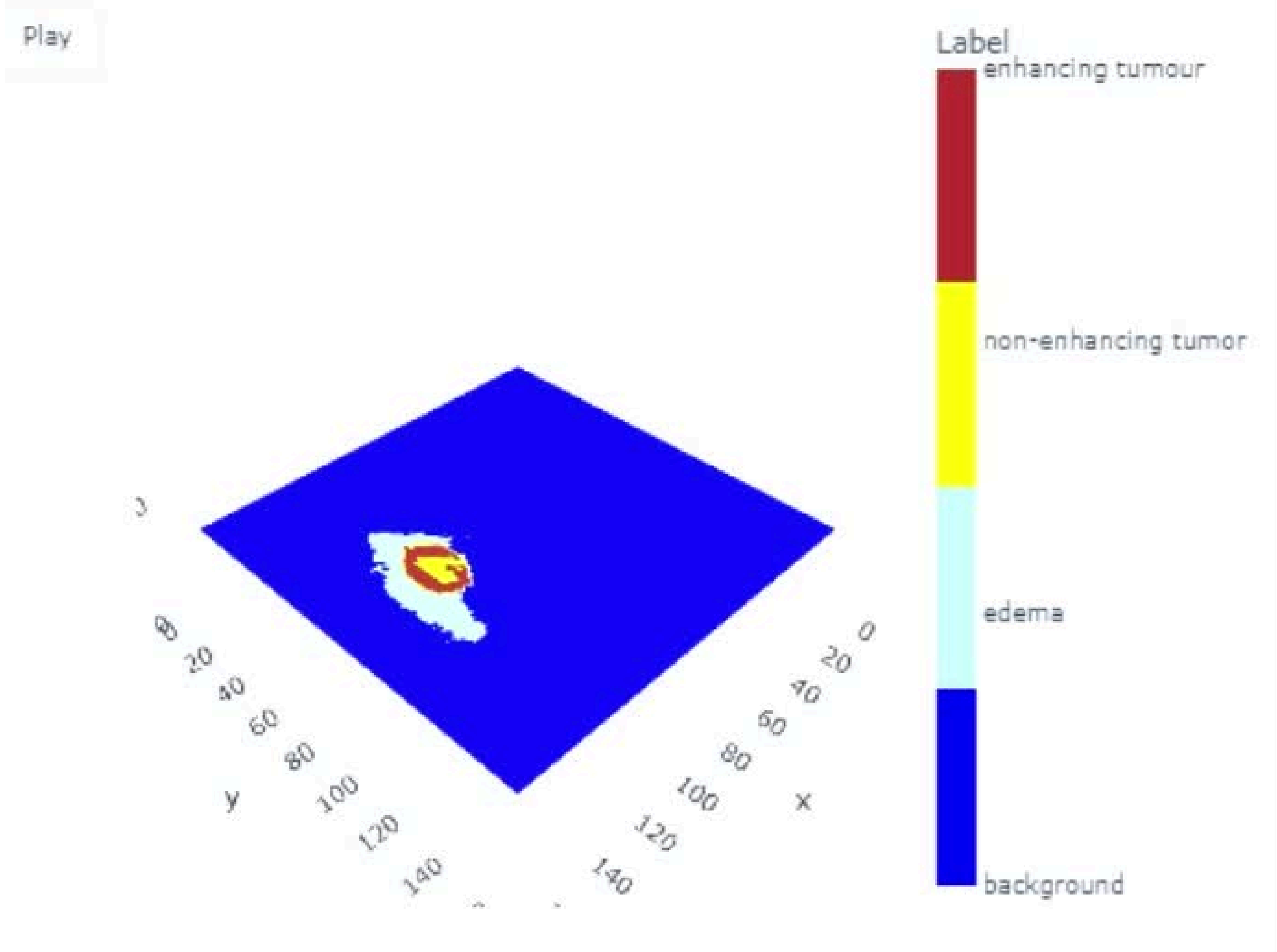
Leonardo Roman
Davide Tortorella

01

3D Medical Image Segmentation

The task of **3D image segmentation** is to **teach** an AI system to automatically assign the **appropriate label** to every **voxel** (the 3D equivalent of a pixel) in a novel 3D volume. Following a **supervised** approach.



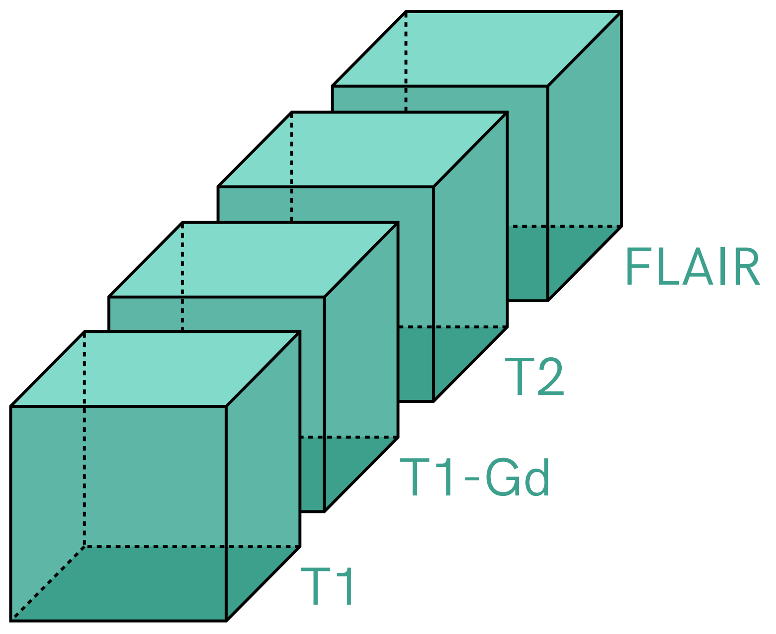


02

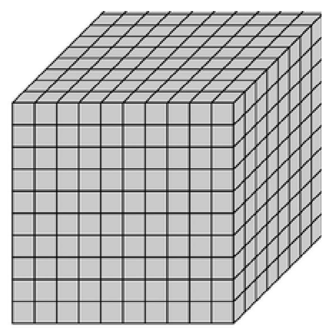
Data Exploration

Characteristics of our data:

- Dimension of a single volume: **240x240x155x4**
- **484** samples
- Each voxel is assigned with a label, **0**: background
  1: edema
  2: non-enhancing tumor
  3: enhancing tumor
- 4 different MRI modalities: native T1-weighted (**T1**), post-Gadolinium contrast T1-weighted (**T1-Gd**), native T2-weighted (**T2**), T2 Fluid-Attenuated Inversion Recovery (**FLAIR**)



Volume 240x240x155x4

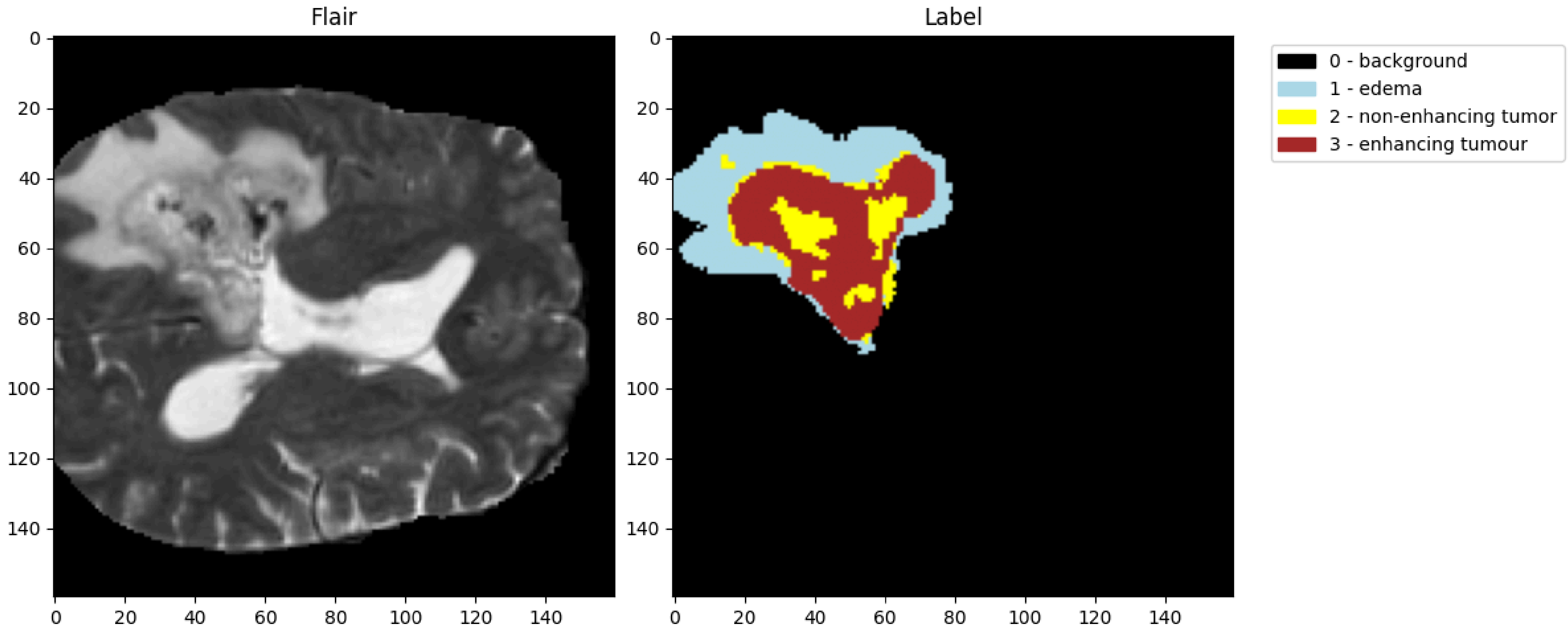


Label 240x240x155

WARNING: Imbalanced Data

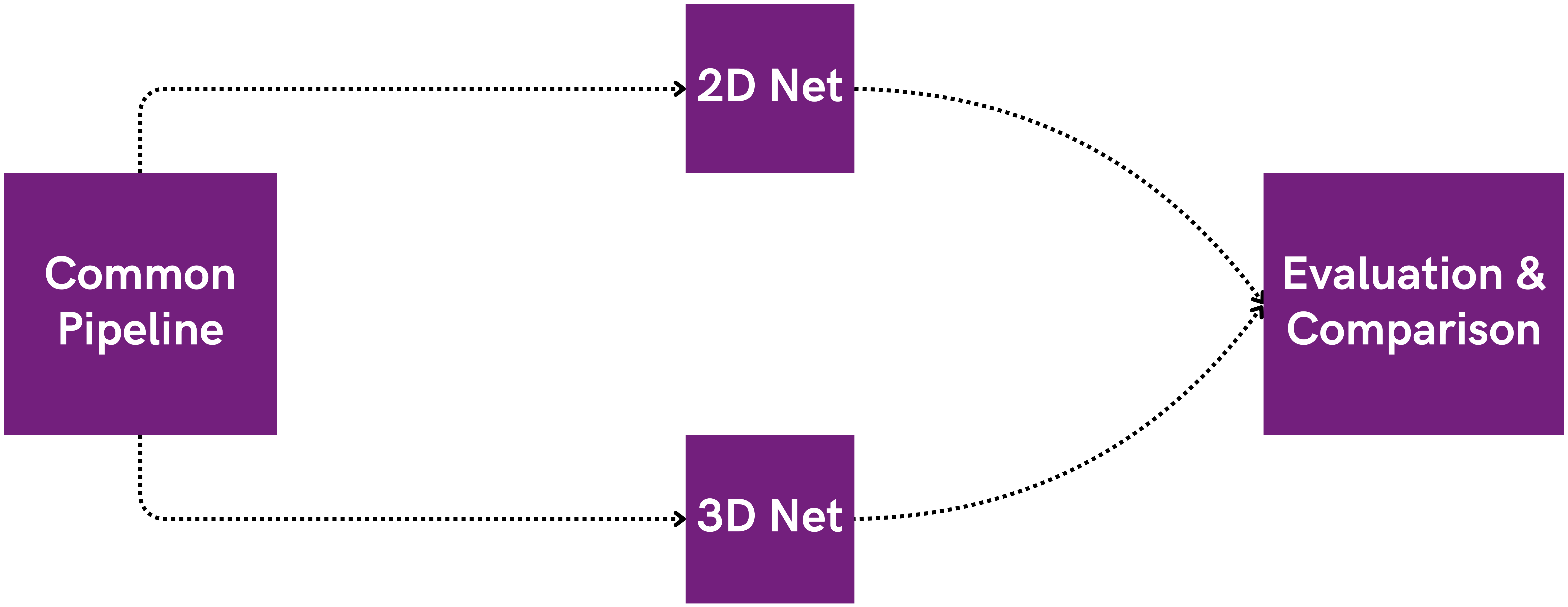
An example

Even when examining slices with the greatest concentration of relevant pixels (labels 1, 2, and 3), it is evident that they are still significantly underrepresented compared to the background.



03

Our Solution(s)


2D Net
Common Pipeline
Evaluation & Comparison
3D Net

Common Pipeline

- **Preprocessing**
  - Cropping
  - Downsampling
  - Normalization

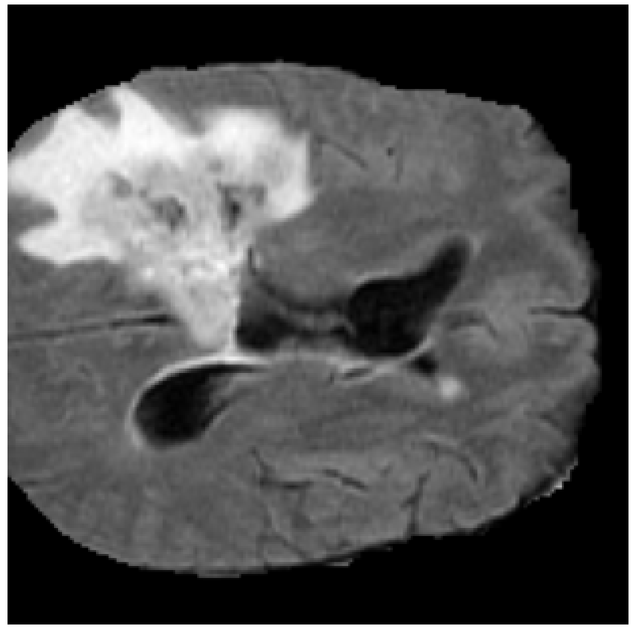
- **Train/Validation/Test** split with **Stratified Sampling**

Train 80%

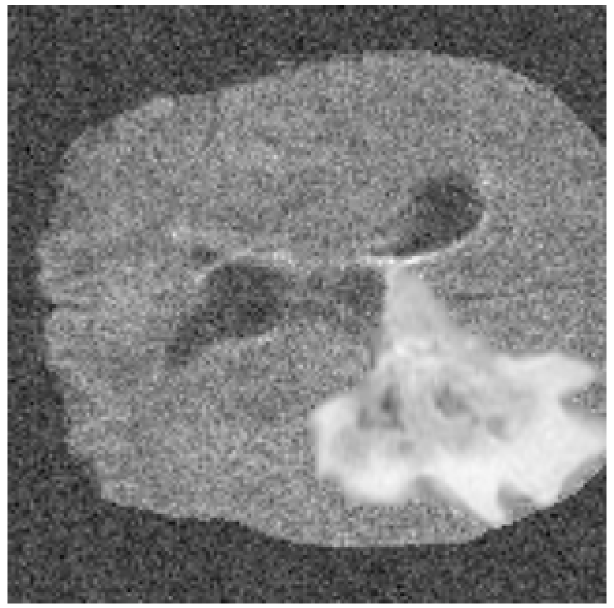
Val 10%

Test 10%

- **Data Augmentation**



Flip
&
Gaussian Noise



Common Pipeline

- ## Loss

As our loss function, we opted for a combination of **Categorical Crossentropy (CCE) Loss** and **Dice Loss**.

- **CCE Loss**, used for multi-class segmentation problems. It penalizes the model more heavily when it is confidently wrong, encouraging more calibrated predictions.

- **Dice Loss**, used in medical image segmentation. It is computed from the Dice coefficient, which measures the overlap between predicted and ground truth regions. To compute it, we one-hot encoded the model's output.

After testing different configurations, we found that an evenly balanced combination of SCCE and Dice Loss effectively handled class imbalance without requiring custom class-weighted functions, a strategy that has also proven effective in other image segmentation studies.

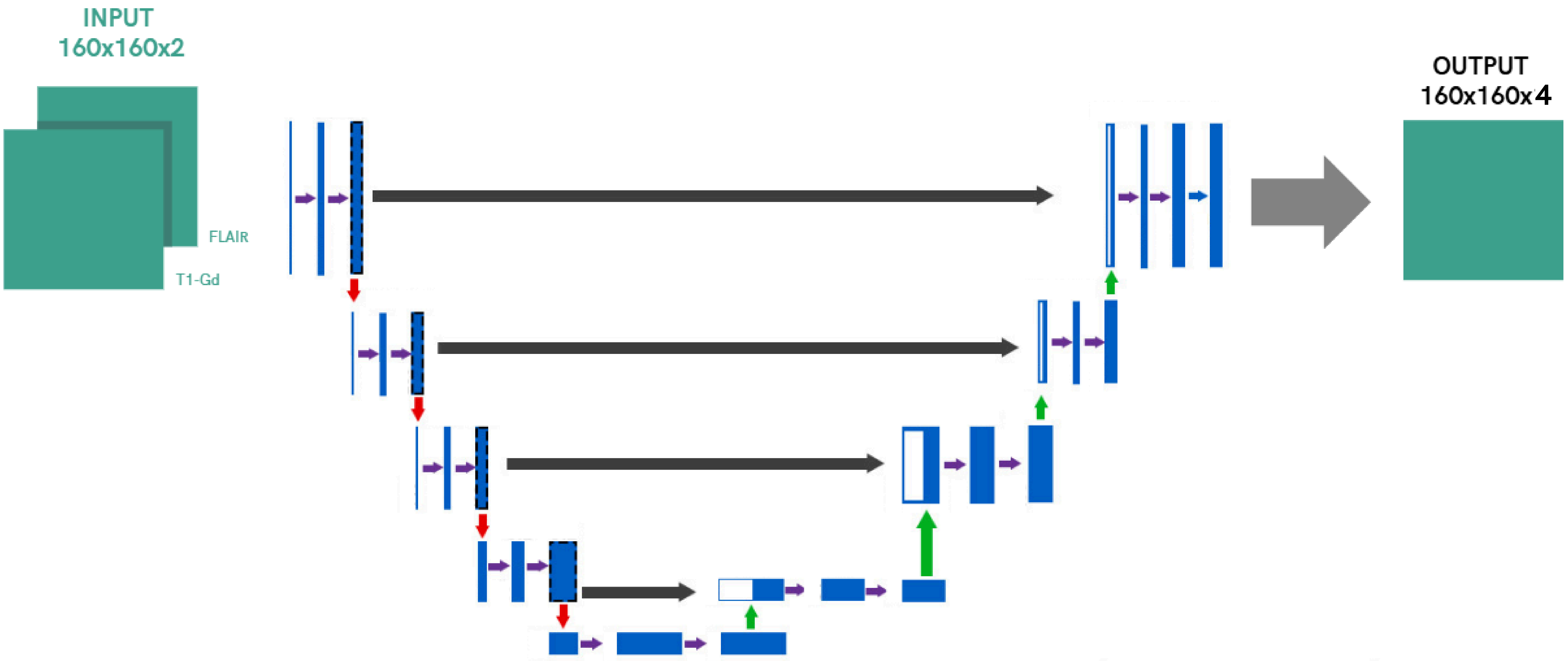
- ## Metrics

As our metric, we chose **Intersection over Union (IoU).**

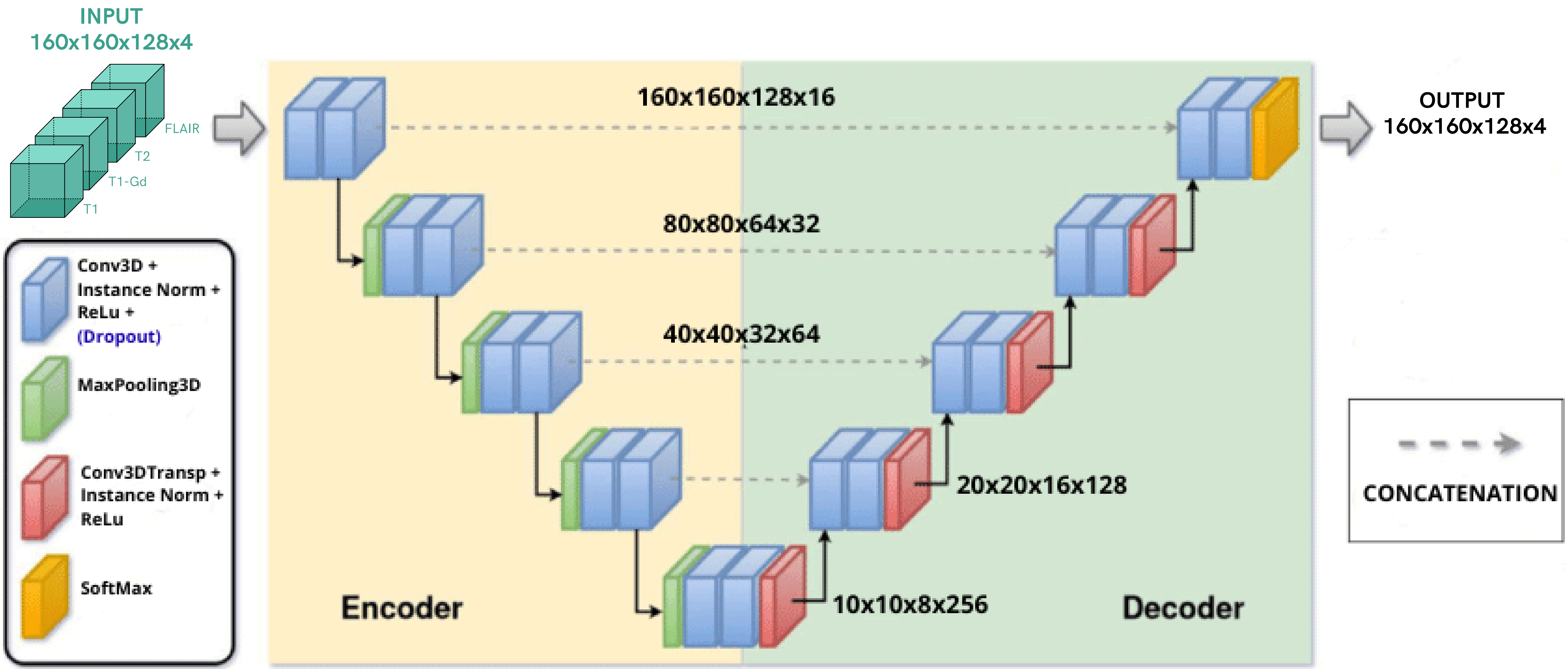
- **IoU**, used in image segmentation tasks, computes the ratio between the area of overlap and the area of union between the predicted segmentation and the ground truth, providing a robust measure of how well the model segments each class.

We chose this metric also because it is widely used in the deep learning literature regarding image segmentation tasks, allowing us to compare our model's performance more directly with results from other studies.

2D Net



3D Net



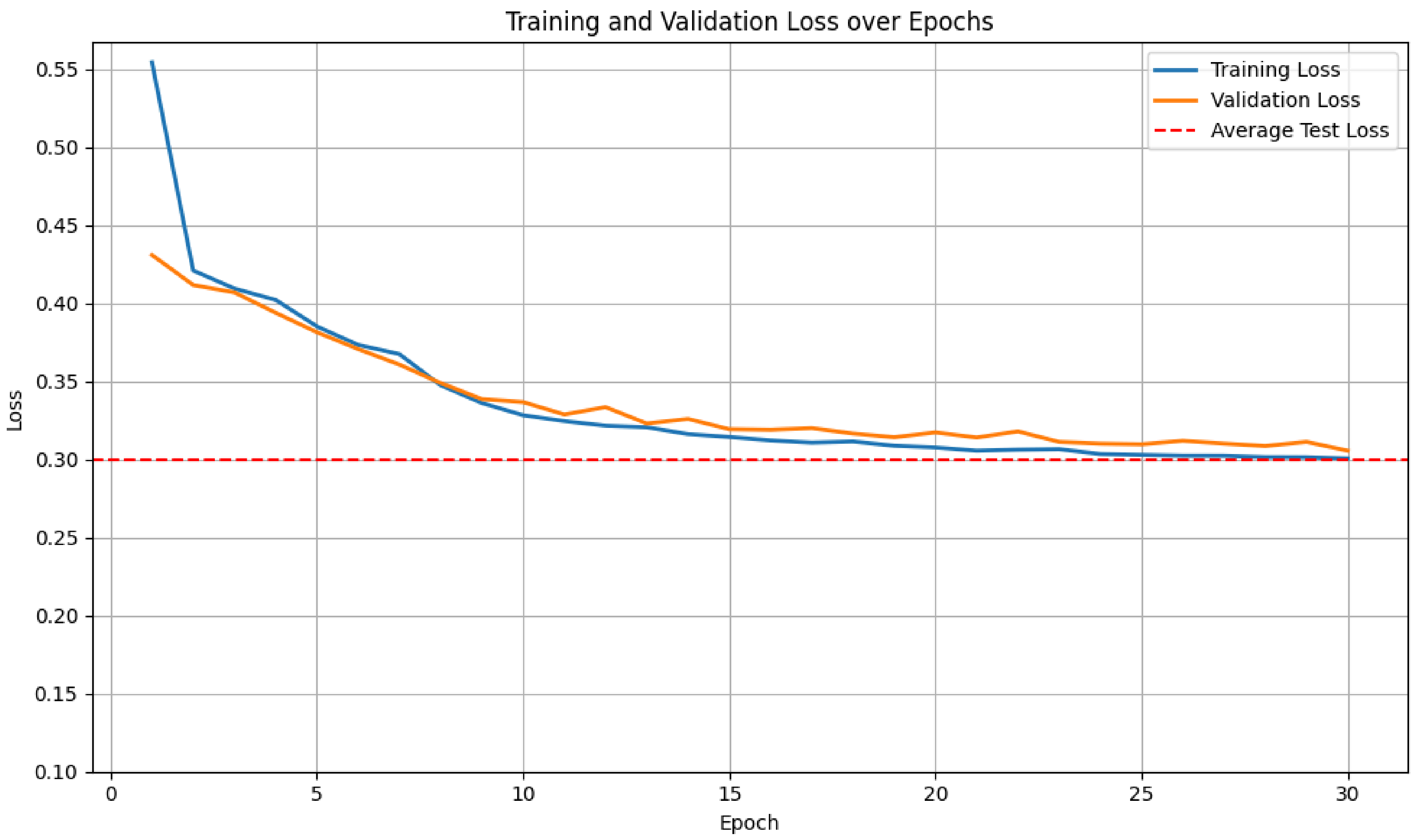
INPUT
160x160x128x4
FLAIR
T2
T1-Gd
T1
Conv3D +
Instance Norm +
ReLu +
(Dropout)
MaxPooling3D
Conv3DTransp +
Instance Norm +
ReLu
SoftMax
160x160x128x16
80x80x64x32
40x40x32x64
20x20x16x128
10x10x8x256
Encoder
Decoder
OUTPUT
160x160x128x4
CONCATENATION

Evaluation & Comparison

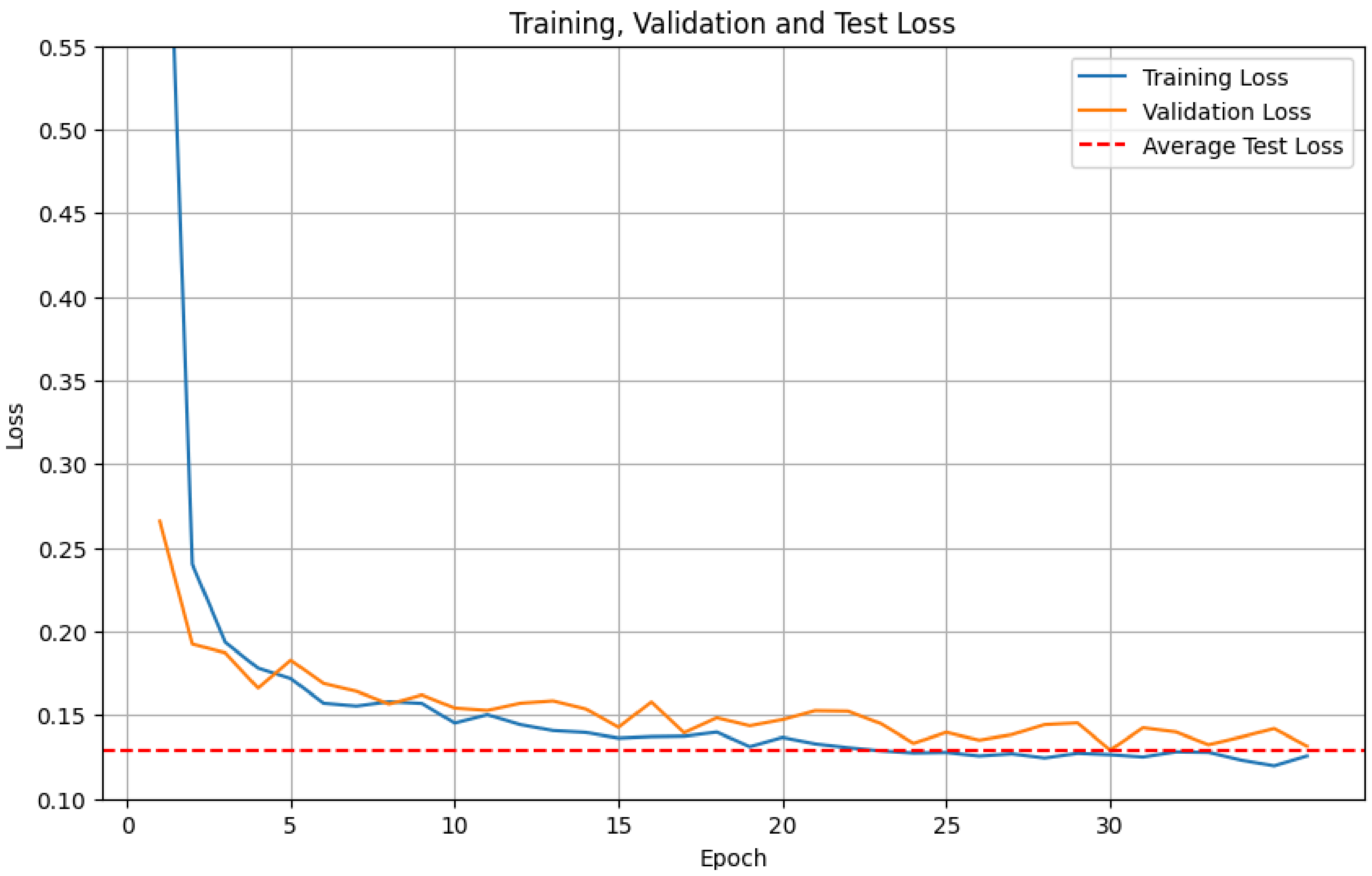
	2D Net	3D Net
Total Params	1,962,532	5,649,652
Batch Size	144	2
Learning Rate	0.001	0.0001
Weight Initializer	*None*	He Uniform
MRI Channels Kept	2	4

Evaluation & Comparison

2D Net



3D Net



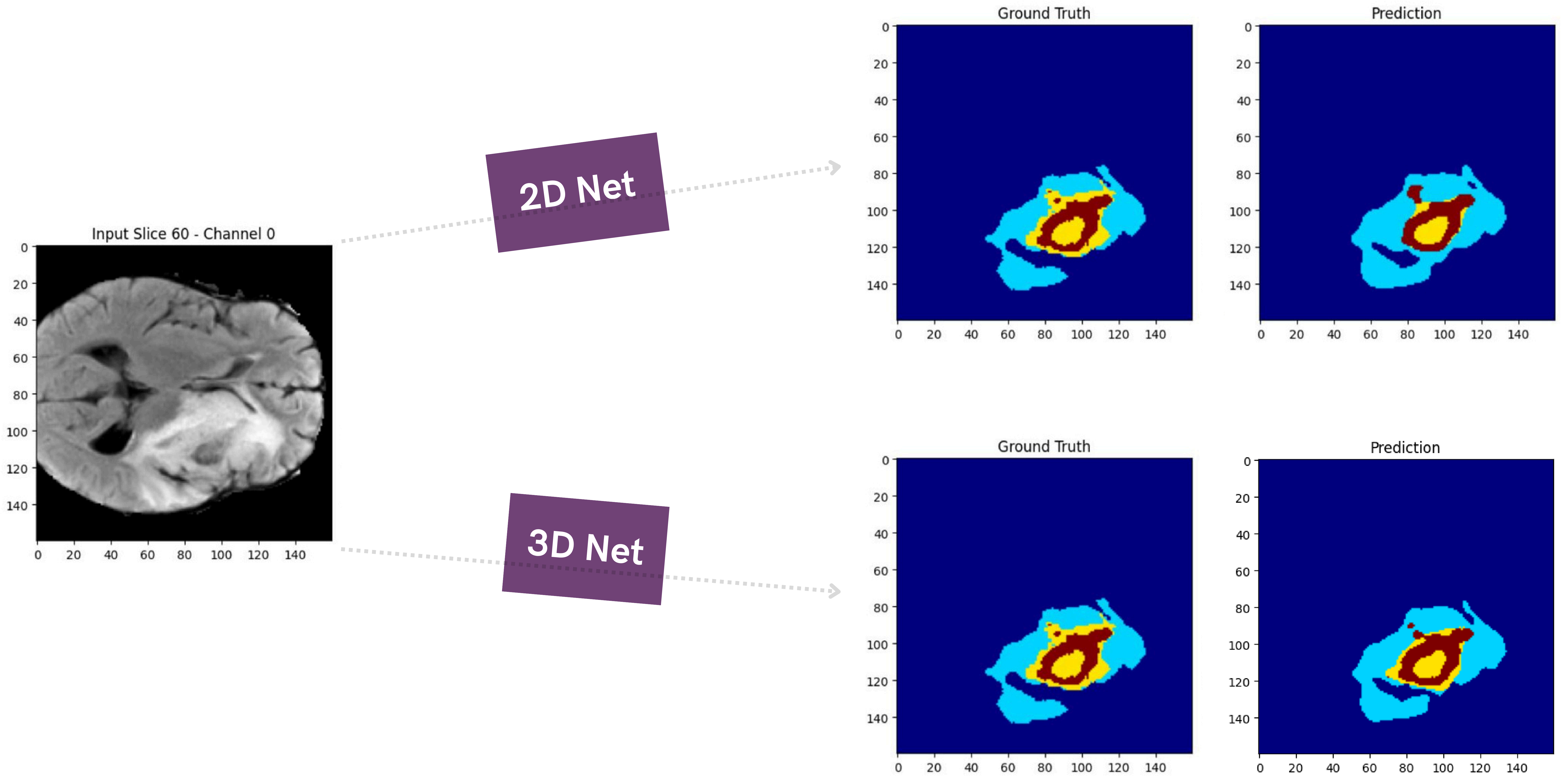
Evaluation & Comparison

2D Net	Precision	Recall	F1-score
0: Background	0.9962	0.9958	0.9960
1: Edema	0.7643	0.8044	0.7839
2: Non enhancing	0.6502	0.5959	0.6219
3: Enhancing	0.8298	0.8130	0.8213

3D Net	Precision	Recall	F1-score
0: Background	0.9962	0.9976	0.9969
1: Edema	0.8236	0.8158	0.8196
2: Non enhancing	0.7468	0.6031	0.6673
3: Enhancing	0.8583	0.8581	0.8581

Recall = TP / (TP+FN)

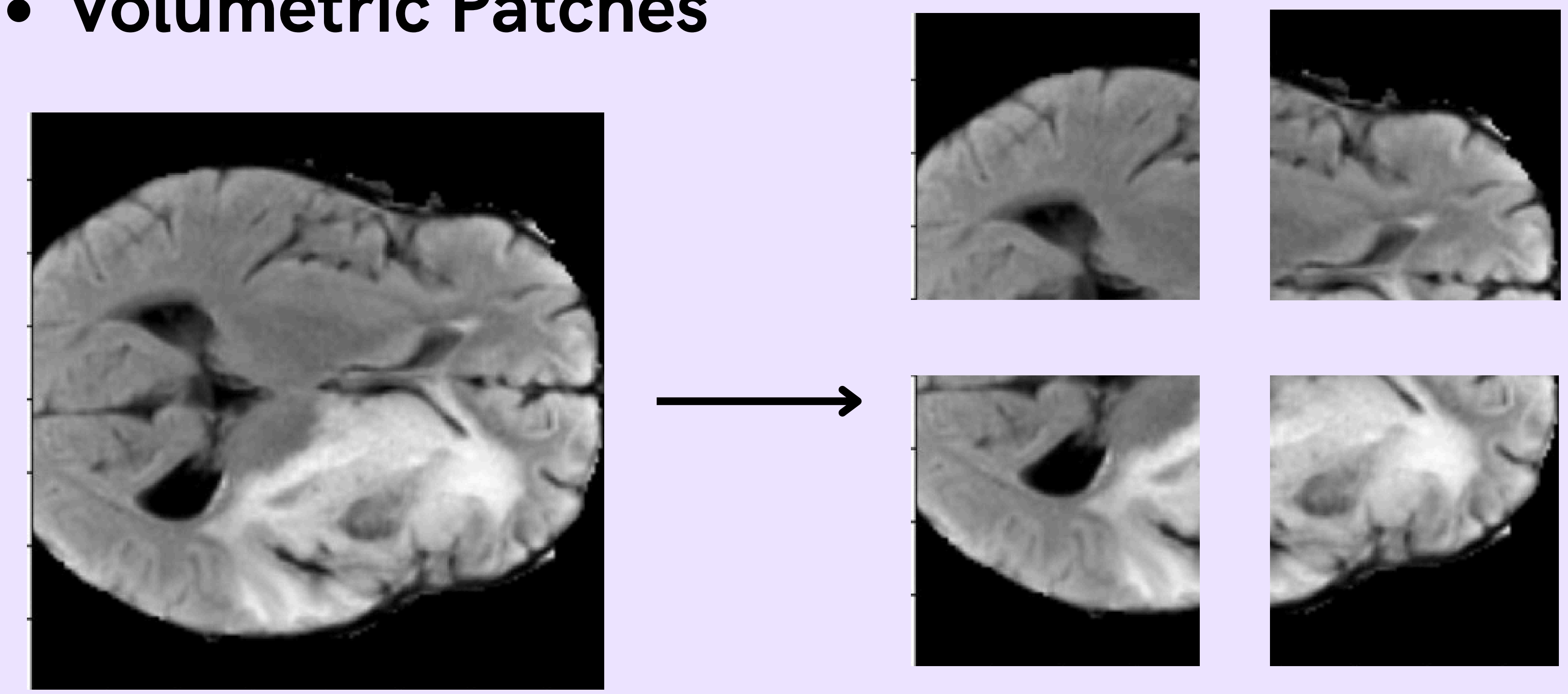
Evaluation & Comparison



04

Further Improvements

- **Volumetric Patches**



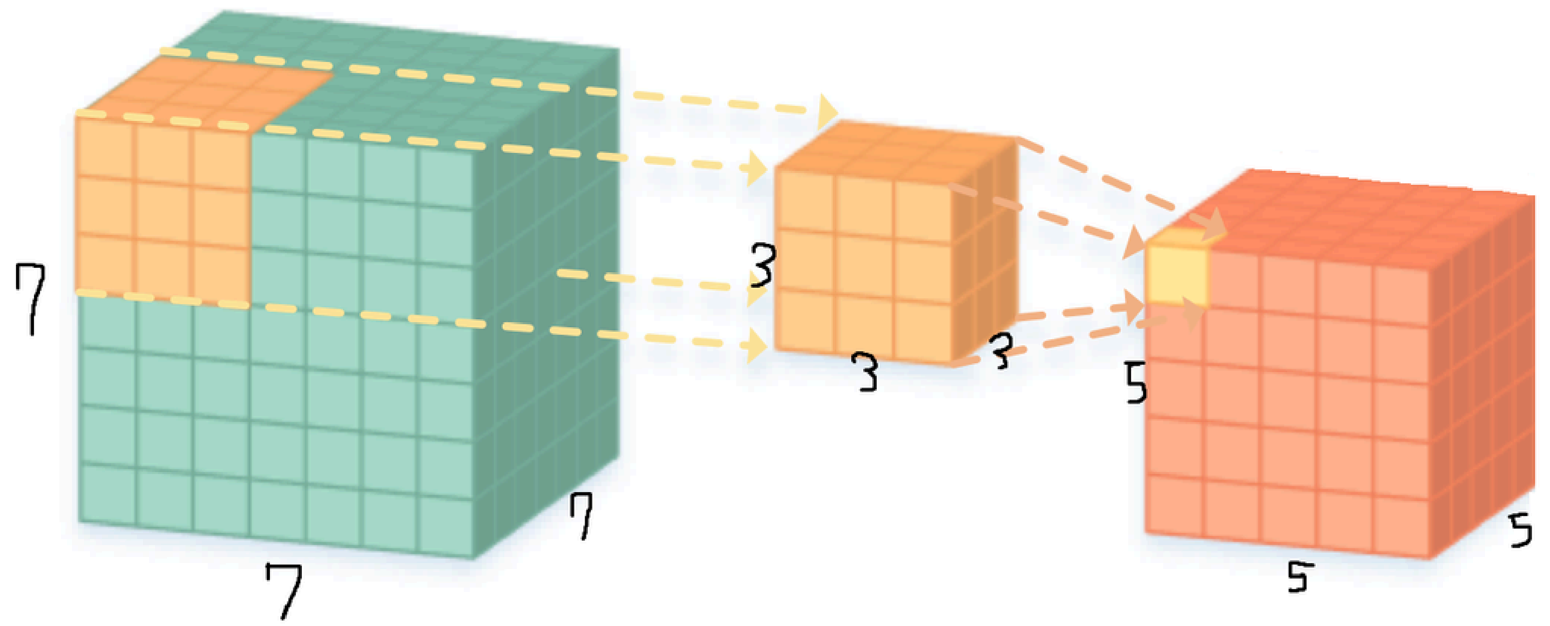
- **Attention U-Net**

- **Test** our model with the discarded "**only background**" samples to simulate a more realistic scenario

Thanks for your attention!

CONV-3D

- For each dimension: **output = (input - k + 2p)/s + 1**



output = (input - k + 2p)/s + 1
<=>
considering p=0, s=1,
5 = (7-3+0)/1 + 1

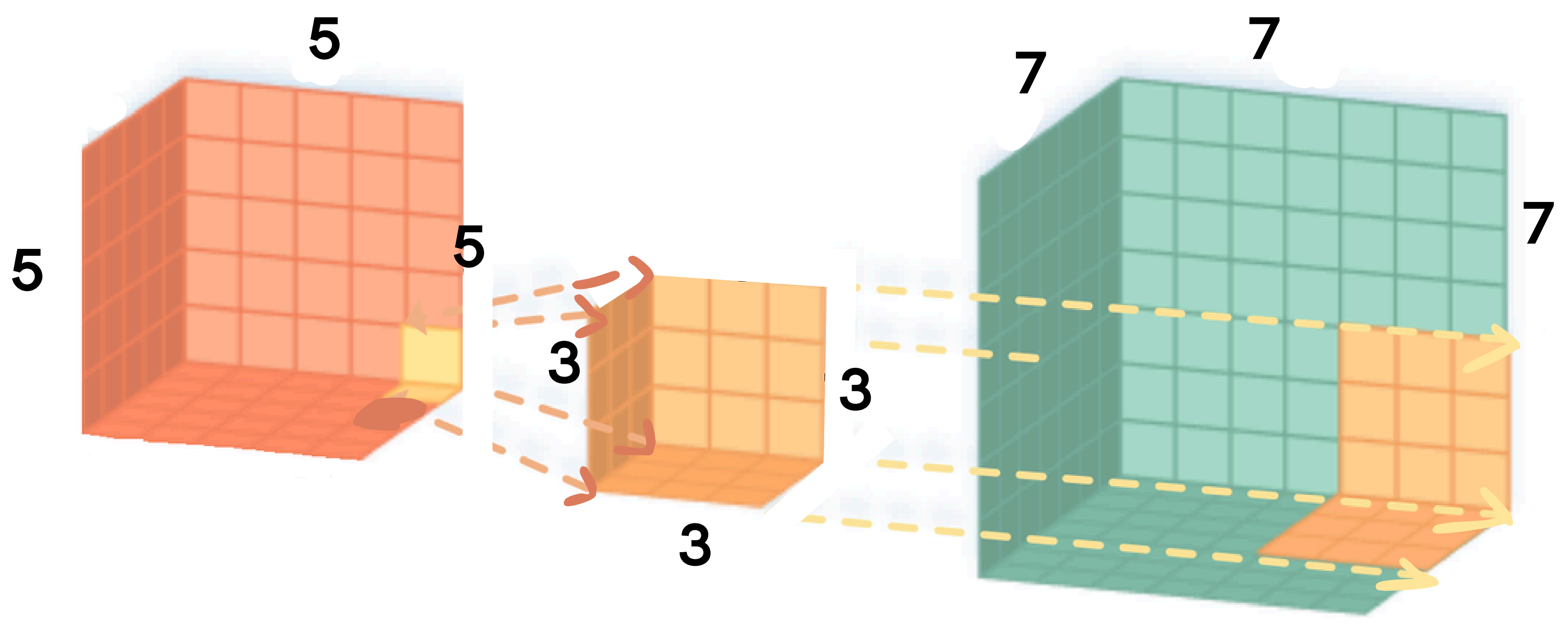
- <u>To maintain the same dimensionality between input and output:</u>
  solve {output = (input - k + 2p)/s + 1
  {input = output
  with respect to p

  0 = -3+2p+1 <=> p=1

  the new input would be 9x9x9

CONV-3D-TRANSPOSE

- For each dimension: **output = (input - 1)*s + k - 2p**



output = (input - 1)*s + k - 2p
<=>
considering p=0, s=1,
7 = (5-1)*1+3-0

- <u>Oss</u>
  stride = 1 but stride != kernel size
  means **overlapping**: values are summed