

AI FOR MEDICAL IMAGING

Multi-institute training for automatic White Matter Hyperintensity segmentation

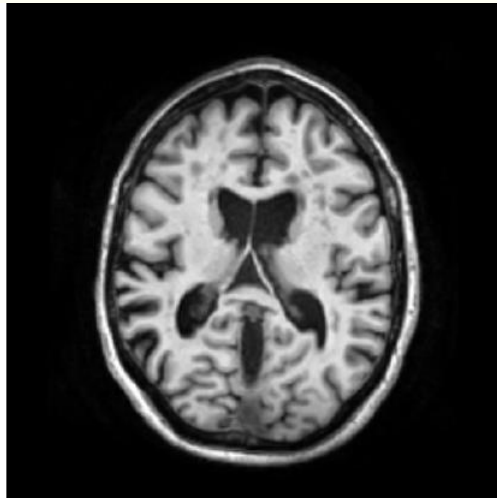
Odysseas Papakyriakou, Paula Castro Ramírez, Jurre Weijer, Abdool Al-Khaledi & Pedro Jesús Álvarez Durán

Outline

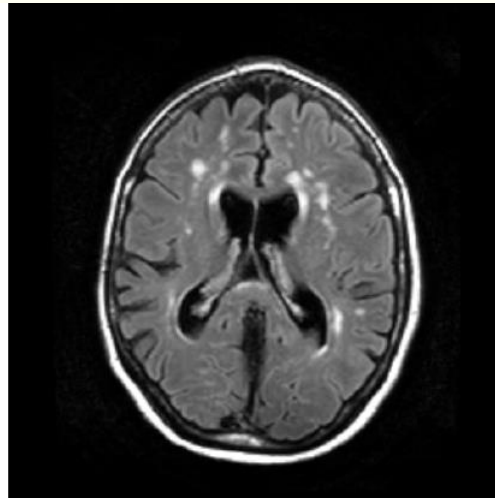
- White Matter Hyperintensity
- Multi-institute training
- Research question and methodology
- Experimental layout
- The U-net model for segmentation
- Data Analysis and Preprocessing
- The Dice Loss
- Results
- Conclusion

White Matter Hyper intensity

- White Matter lesions (T1 + Flair in our case).
- Presents as hyper/hypo intense regions on MR images.
- Can be a result of: Ischemia, micro-hemorrhages, gliosis, damage to small blood vessels, etc.
- Convention involved manual segmentation



(a) T1-weighted image



(b) FLAIR image



(c) Manual WMH segmentation

Fig. 1. Example brain MR images of a subject with white matter hyperintensities (WMH) of presumed vascular origin. On the T1-weighted image (a), WMH show as hypointense regions within the white matter. On the FLAIR image (b), WMH are clearly visible as hyperintense regions within the white matter. The corresponding manual WMH segmentation is shown in (c).

Kuijf, Hugo J et al. "Standardized Assessment of Automatic Segmentation of White Matter Hyperintensities and Results of the WMH Segmentation Challenge." *IEEE transactions on medical imaging* vol. 38,11 (2019): 2556-2568.
doi:10.1109/TMI.2019.2905770

Multi Institute training

- The process of reconciling data from different institutions.
- The idea is to train a model on data from different sources/institutions to increase generalizability.
- When data from different sources are available, different combinations of training data can yield different results.
- Data acquired from 3 institutes:
 - University Medical Center (UMC) Utrecht.
 - VU University Medical Centre (VU) Amsterdam.
 - National University Health System (NUHS) in Singapore.



Research question

- How does multi-institute training change the segmentation performance of white matter hyperintensity?

Methodology

- Develop and train a U-net model on different combinations of data from the three institutes. Test the generated segmentations on scans from each institute and compare the model's performance.
- The model is evaluated using the dice loss ($1 - \text{dice coefficient}$).

Experimental layout

	Train set			Test set		
Set name	Utrecht	Amsterdam	Singapore	Utrecht	Amsterdam	Singapore
Utrecht	16	0	0	4	0	0
Amsterdam	0	16	0	0	4	0
Singapore	0	0	16	0	0	4
Utr + Ams	8	8	0	2	2	0
Utr + Sing	8	0	8	2	0	2
Ams + Sing	0	8	8	0	2	2
Utr+Ams+Sing	5	5	5	1	1	1
All	16	16	16	4	4	4

- The validation set is automatically created using a 0.1 "train_test_split", split at a patient level.

The U-net model (1)

- Originally developed by Ronneberger, Fischer, and Brox in 2015
- Particularly useful for high resolution segmentation in biomedical scans

1. Encoding path (downsampling, contracting path)

-> several 3x3 convolutional layers with ReLU activation functions and 2x2 max-pooling with stride 2 for downsampling

-> progressively reduce the spatial dimensions while doubling the number of feature maps to learn distinctive features at different scales

2. Bottleneck

-> a compressed representation of the input data, ideally capturing only useful features

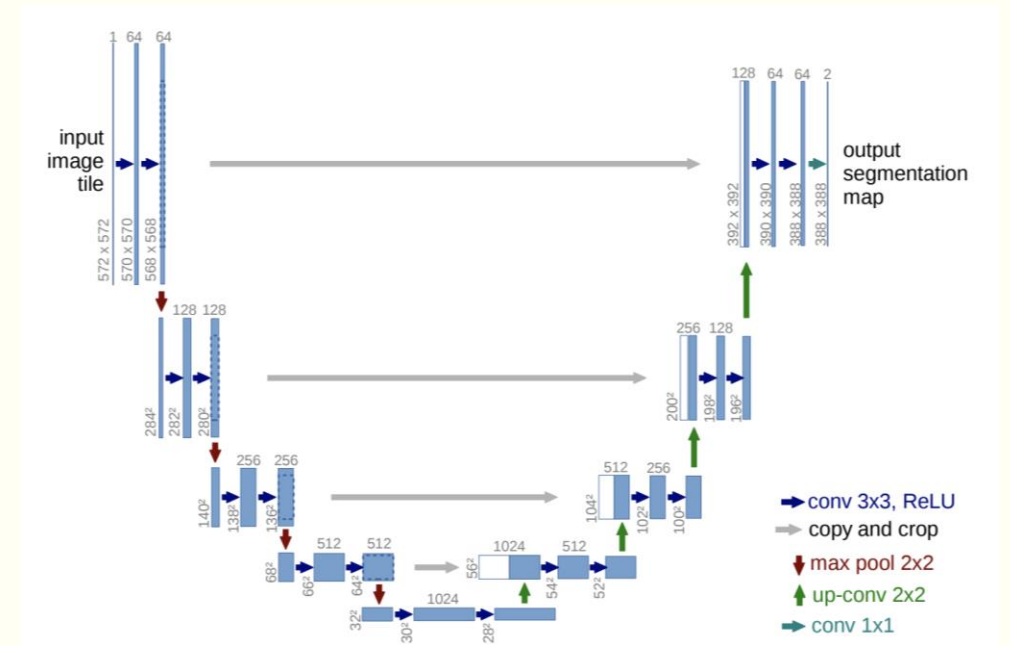


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Image from the original paper by Ronneberger et al., 2015. Link [here](#)

The U-net model (2)

- 3. Decoding path (upsampling, expansive path)
 - -> 2x2 upsampling layers used to increase the feature maps' spatial resolution with transpose convolutions while halving the number of feature maps
 - -> concatenation (skip connections) with the corresponding feature maps from the encoding path to preserve spatial information and cropping to account for the loss of border pixels in every convolution
 - -> followed by two 3x3 convolutions each with ReLU activation
- 4. Output
 - -> 1x1 convolution to map the learned features to the desired number of output maps, representing the segmentation mask

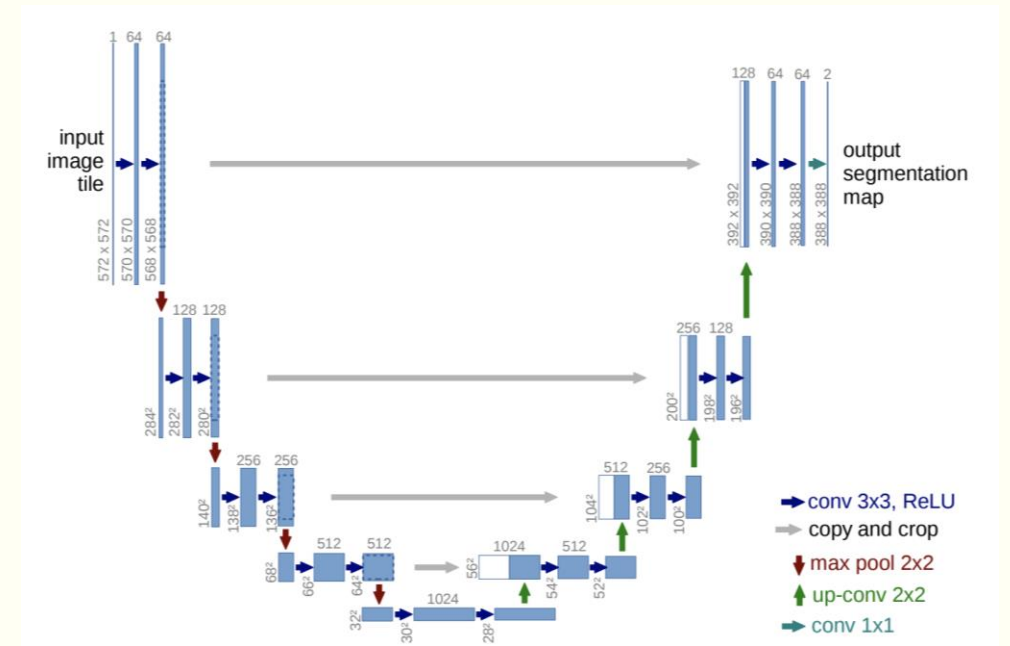
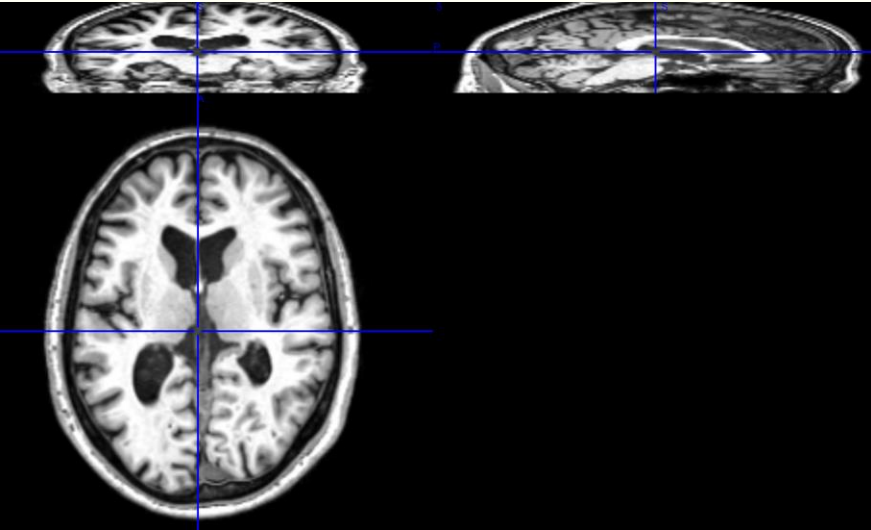


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Image from the original paper by Ronneberger et al., 2015. Link [here](#)

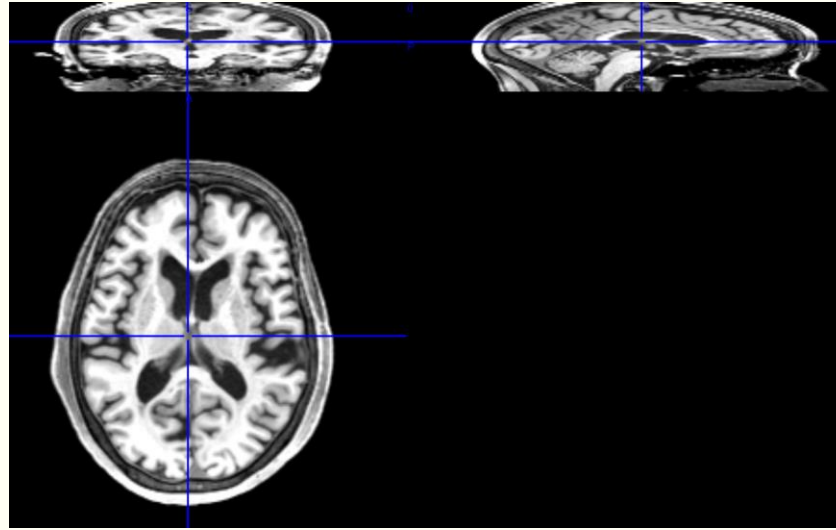
Data Analysis

Utrecht



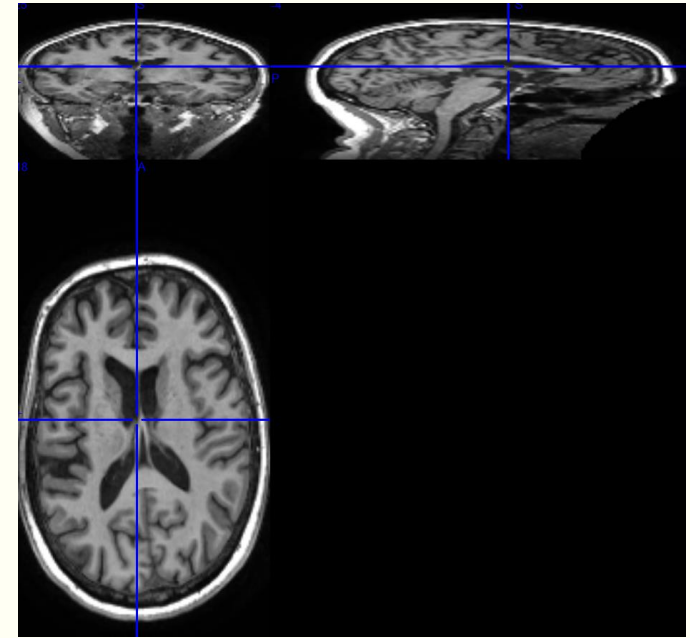
48x240x240

Singapore



48x256x232

Amsterdam



83x256x132

Data Analysis – Resolution

Utrecht

Dimension	Length	Spacing
I Space	<input type="text" value="240"/>	<input type="text" value="0,9583"/>
J Space	<input type="text" value="240"/>	<input type="text" value="0,9583"/>
K Space	<input type="text" value="48"/>	<input type="text" value="3,0000"/>

Singapore

Dimension	Length	Spacing
I Space	<input type="text" value="232"/>	<input type="text" value="1,0000"/>
J Space	<input type="text" value="256"/>	<input type="text" value="1,0000"/>
K Space	<input type="text" value="48"/>	<input type="text" value="3,0000"/>

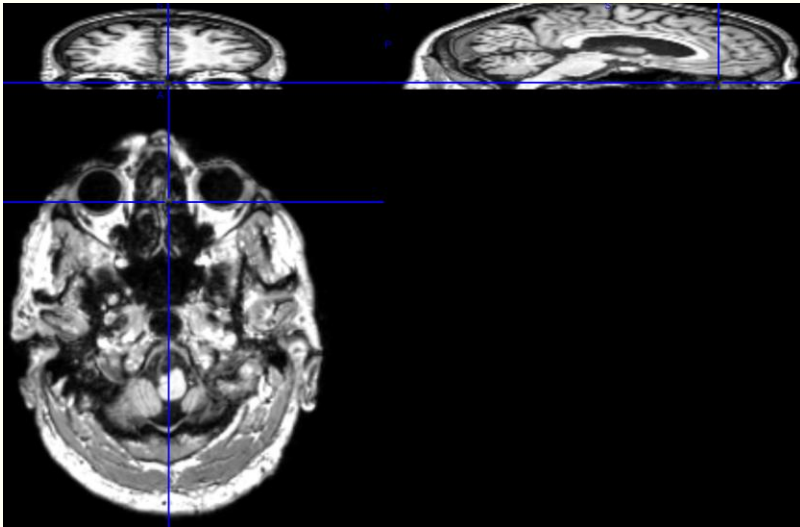
Dimension	Length	Spacing
I Space	<input type="text" value="256"/>	<input type="text" value="1,0000"/>
J Space	<input type="text" value="232"/>	<input type="text" value="1,0000"/>
K Space	<input type="text" value="48"/>	<input type="text" value="3,0000"/>

Amsterdam

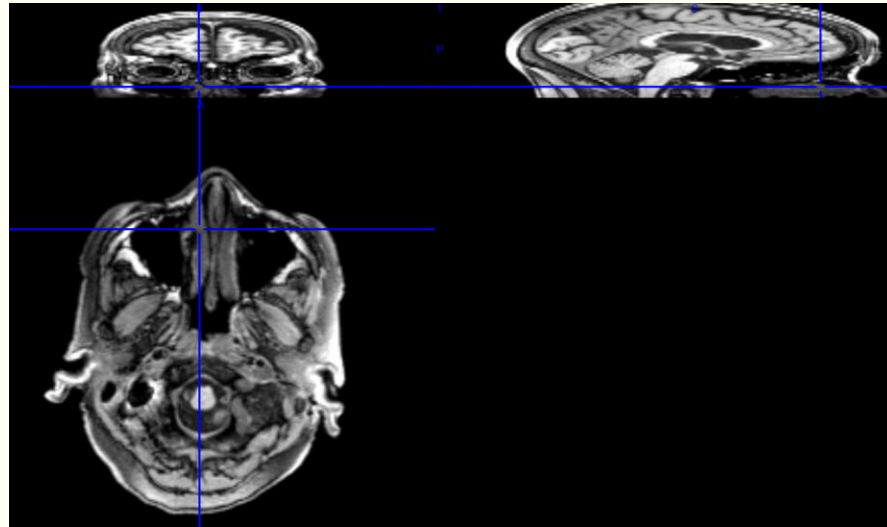
Dimension	Length	Spacing
I Space	<input type="text" value="132"/>	<input type="text" value="1,2000"/>
J Space	<input type="text" value="256"/>	<input type="text" value="0,9766"/>
K Space	<input type="text" value="83"/>	<input type="text" value="3,0000"/>

Data Analysis – Acquisition Method

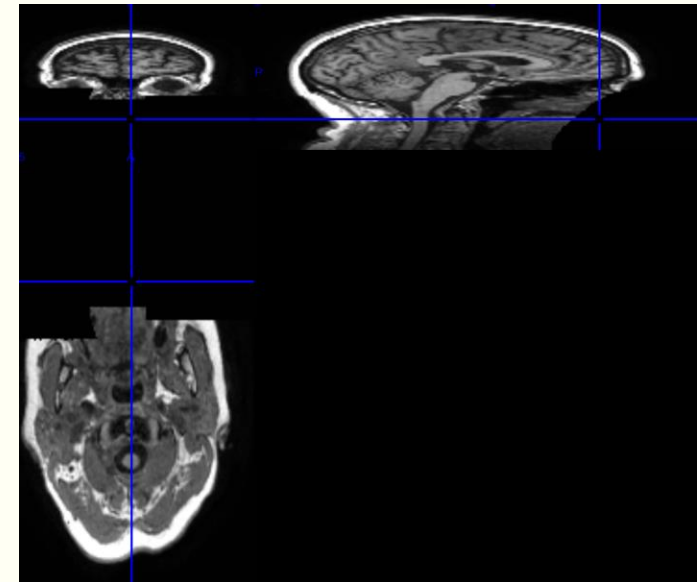
Utrecht



Singapore

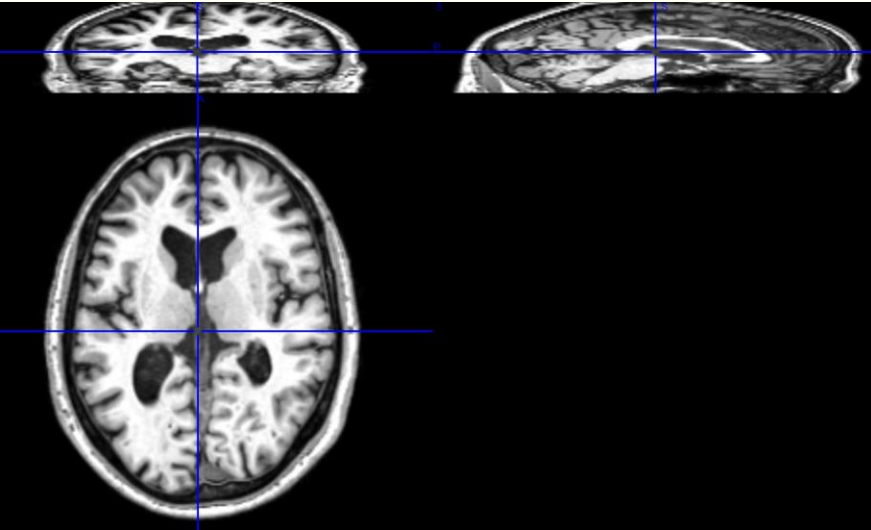


Amsterdam



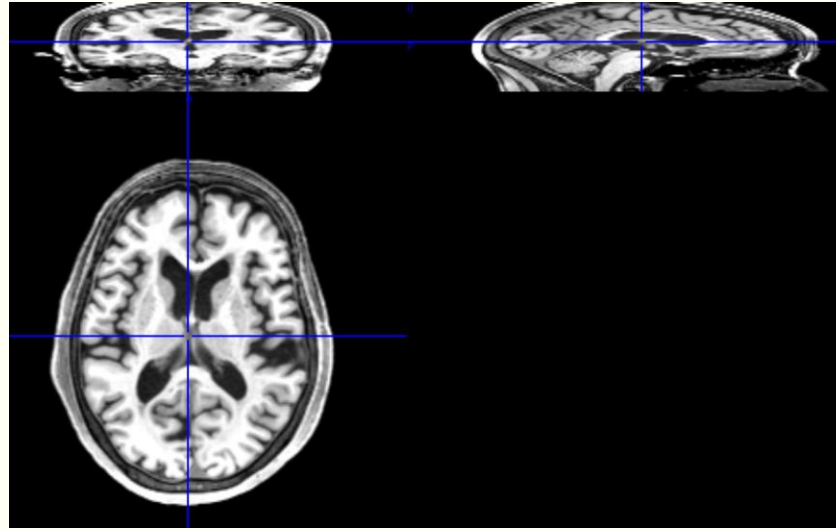
Preprocessing

Utrecht



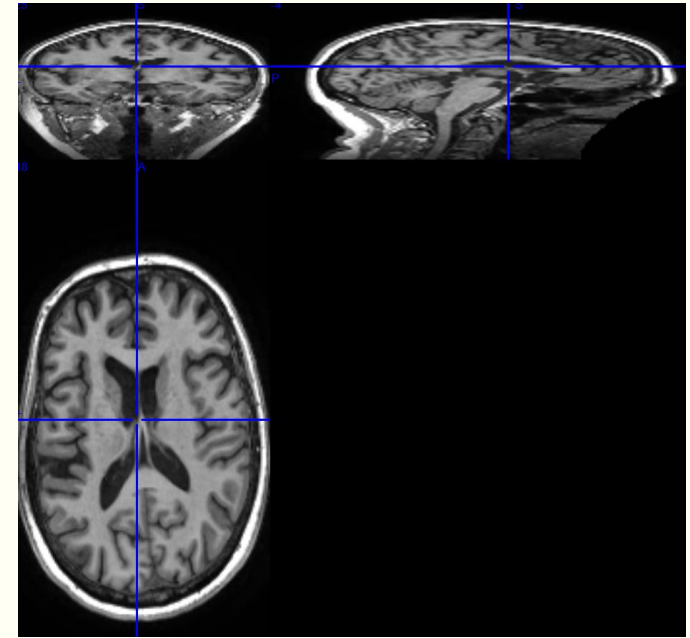
48x256x256

Singapore



48x256x256

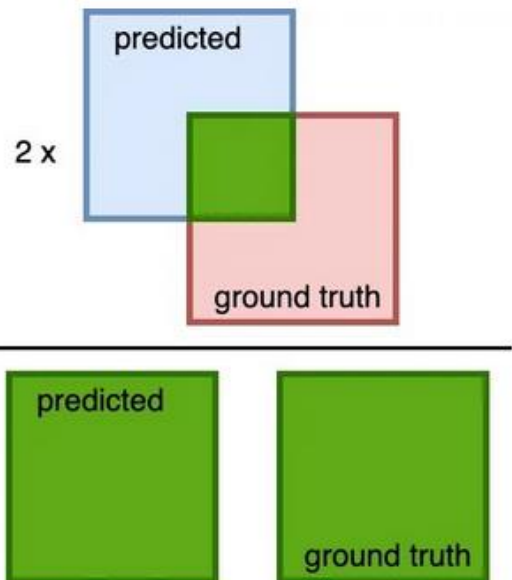
Amsterdam



83x256x256

The Dice loss

- The Dice loss is a loss function used in image segmentation tasks.
- It measures the similarity between the predicted segmentation and the ground truth segmentation.
- The Dice coefficient ranges between 0 (no similarity) and 1 (perfect similarity).
- The Dice loss is defined as 1- Dice coefficient and can be used as an evaluation metric.

$$\text{Dice coefficient} = \frac{2 \times \text{area of overlapped (green)}}{\text{total area (green)}} = \frac{\text{predicted} \cap \text{ground truth}}{\text{predicted} \cup \text{ground truth}}$$


Results, showing the dice loss (1 – dice coefficient)

- The closer the dice loss is to 0, the greater the similarity between the generated and the human segmentation

Train/Test	Utrecht	Ams	Sing	Utr/Ams	Ams/Sing	Utr/Sing	Utr/Ams/Sing
Utrecht	0.160	0.100	0.194	0.080	0.149	0.174	0.114
Ams	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Sing	0.267	0.084	0.181	0.117	0.137	0.224	0.130
Utr/Ams	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Ams/Sing	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Utr/Sing	0.155	0.054	0.095	0.063	0.091	0.116	0.054
Utr/Ams/Sing	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Utr/Ams/Sing/all	0.267	0.083	0.180	0.116	0.136	0.223	0.129

Conclusion



Train/Test	Utrecht	Ams	Sing	Utr/Ams	Ams/Sing	Utr/Sing	Utr/Ams/Sing
Utrecht	0.160	0.100	0.194	0.080	0.149	0.174	0.114
Ams	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Sing	0.267	0.084	0.181	0.117	0.137	0.224	0.130
Utr/Ams	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Ams/Sing	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Utr/Sing	0.155	0.054	0.095	0.063	0.091	0.116	0.054
Utr/Ams/Sing	0.267	0.083	0.180	0.116	0.136	0.223	0.129
Utr/Ams/Sing/all	0.267	0.083	0.180	0.116	0.136	0.223	0.129

- The model trained on Utrecht and Singapore data seems to be the most generalizable
- The Amsterdam dataset seems to yield the best results across all models
- For a lot of models trained with different data the results are similar across different test sets
 - eg. When trained with Ams; Sing; Utr/Ams; Ams/Sing; Utr/Ams/Sing

Discussion

- Why do all models predict the Amsterdam data so well?
 - Maybe taking out personal information(nose, eyes, etc.) reduces noise
- Why does a combination of data from Utrecht and Singapore lead to such a better performance?