Automated Detection of MR Imaging Biomarkers of Cerebral Small Vessel Disease

MedDigit - Medicine and Digitalization, Otto-von-Guericke-Universität Magdeburg, Universitätsklinik für Neurologie
Professor: PD Dr.-Ing habil Steffen Oeltze-Jafra
Supervisor: Alessandro Sciarra and Max Dünnwald
Project Team: Harish Kumar Harihara Subramanian; Logesh Babu Radhakrishnan; Balakrishnan Ramakrishnan; Abhivanth Murali

1. INTRODUCTION

This is an MR imaging biomarker detection of cerebral small vessel disease project using deep learning neural networks. Cerebral small vessel disease (SVD) is an umbrella term covering a variety of abnormalities related to small blood vessels in the brain. Because most brain tissue appears white on MRIs, these abnormalities were historically referred to as "white matter changes." mostly seen in the aging individuals because of brain aging. To confirm the existence of this disease, there should be an existence of at least one of these markers *White Matter Hyperintensities, Lacunes, Perivascular Space, Cerebral Microbleeds, Recent Small Subcortical Infarcts*.

The presence of the above-mentioned lesions is associated with various neurological disorders such as dementia, stroke, hemorrhaging. Manual detection of these lesions from MRI scans is found to be a time consuming and laborious work for analysts. Another drawback is a high variance in inter-rating and intra-rating agreements. Thus automatic detection of these lesions is necessary and deep learning is the most accurate technique used in recent days employed in biomedical applications. Two*(WMH, CMB)* out of 5 markers are selected as part of this project and implemented based on the state of the art papers.

2. PAPERS, METHODS, DATASETS, RESULTS OVERVIEW

The White Matter Hyperintensities detection method is implemented based on *Fully Convolutional Network Ensembles for White Matter Hyperintensities Segmentation in MR Images (Hongwei Li et al).* The technique used in this paper is considered to be state of the art in WMH segmentation as it provides good measure over different evaluation metrics such as *Dice Similarity Coefficient (DSC), Hausdorff distance, Average volume difference(AVD), Sensitivity score(Recall), and F1-score*. It incorporates a slightly alternate variation of FCN, in combination with U-Net to detect WMH in MRI scans. An ensemble model obtained by tuning hyperparameters is employed for classification and this helps to avoid overfitting. The training dataset utilized is a publicly available dataset acquired from the event "MICCAI WMH Segmentation Challenge" https://wmh.isi.uu.nl/data/. The dataset consists of 60 images (in total) acquired from 3 different scanners. It consists of 3D T1-weighted images and 2D FLAIR images for each subject. These training images are obtained from three different scanners in different hospitals & countries (UMC Utrecht, VU Amsterdam, and NUHS Singapore). Thus cross-scanner and cross-subject accuracies can be evaluated. Thus cross-scanner and cross-subject accuracies are also evaluated. Using this method we have achieved the results close to the paper with average scores of 9 held out test subjects as 0.78 DSC, 23.18 AVD, 0.92 Recall, and 0.72 F1-score.

The cerebral microbleeds detection method is implemented based on *Automatic Detection of Cerebral Microbleeds From MR Images via 3D Convolutional Neural Networks(Qi Dou et al).* The technique accurately and efficiently detects CMBs from volumetric brain SWI data, using a robust and efficient method by leveraging 3D CNNs. This method consists of two stages that are designed in a cascaded manner. The first stage is the screening stage, in which a small number of candidates are retrieved using a novel 3D fully convolutional network (3D FCN) model. The second stage is the discrimination stage, where the candidates

obtained from the screening stage are carefully distinguished with a 3D CNN discrimination model. This method achieved better results than the state-of-the-art methods in terms of sensitivity, precision, and false-positive rate. The model is trained using the *Alzheimer's Disease Neuroimaging Initiative (ADNI)* dataset consists of 501 subjects with the trace of CMBs.

3. WHITE MATTER HYPERINTENSITIES IMPLEMENTATION

3.1 Data preparation & Pre-processing

We considered UMC Utrecht (240 x 240 x 48), NUHS Singapore (252 x 232 x 48), and VU Amsterdam (132 x 256 x 83) datasets for training the model. Each dataset has 20 subjects with FLAIR & T1 modalities and also the respective ground truth images. In the first step, the dimensions of the image are set to equal using the center cropping technique only in the XY direction to 200 x 200. As the VU Amsterdam dataset has dimension less than 200 in the x-direction, the dimension is increased to 200 by zero-padding the images before center cropping. Then the images are sliced axially in the z-direction. In the second step, Gaussian normalization was employed to normalize the intensity distribution of the slices. This includes three steps. Firstly, the thresholds were empirically set to 70 for FLAIR and 30 for T1 respectively. Secondly, the holes in the sliced images are filled using morphological operations. Finally, the sliced images are normalized between the range -1 to +1.

3.2 Architecture

In this model, two convolutional layers are repeatedly employed, each followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for downsampling. At the final layer, a 1×1 convolution is used to map each 64-component feature vector to two classes. In total the network contains 19 convolutional layers. The network is built as per the below table.

Optimizer	Adam
Learning Rate	0.0002
Loss	Dice Loss Coefficient
Batch Size	16
Training Epochs	100

Table 1: Hyper Parameters for U-Net network architecture

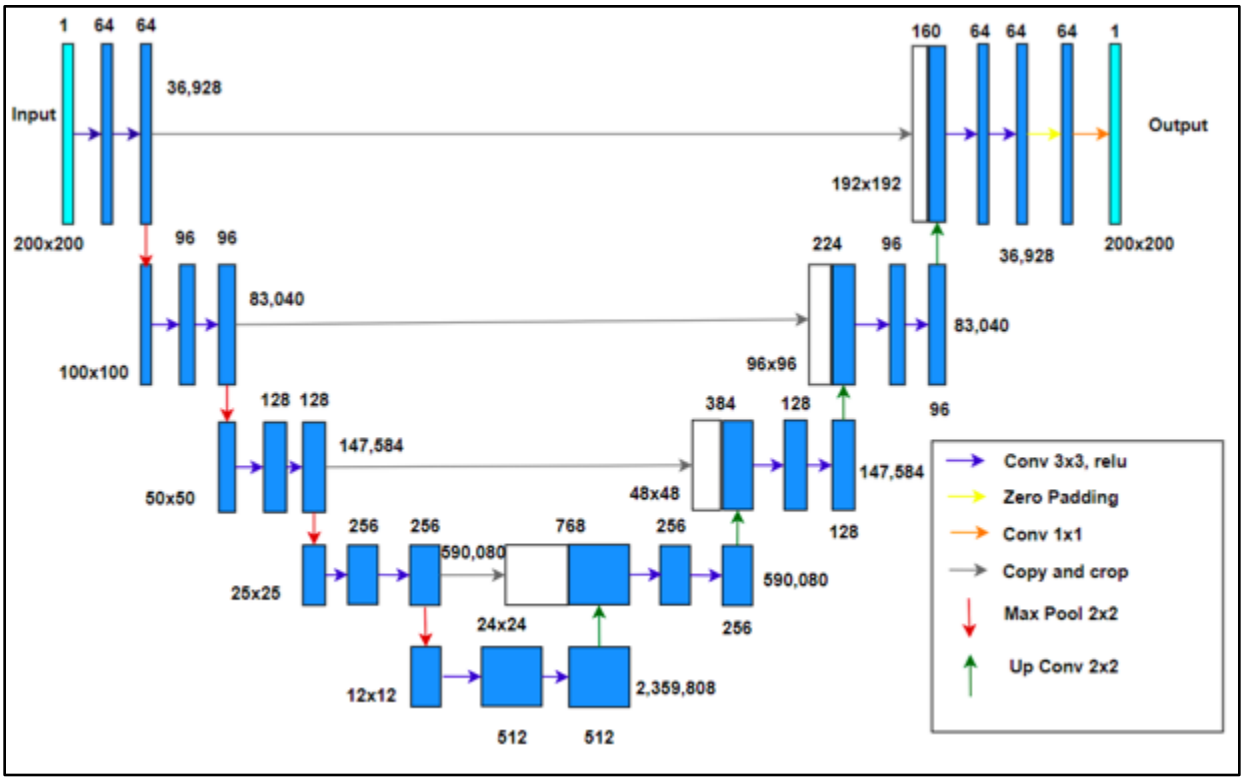

Figure 1: U-Net architecture with dimensions and trainable parameters

3.2 Steps to perform the execution

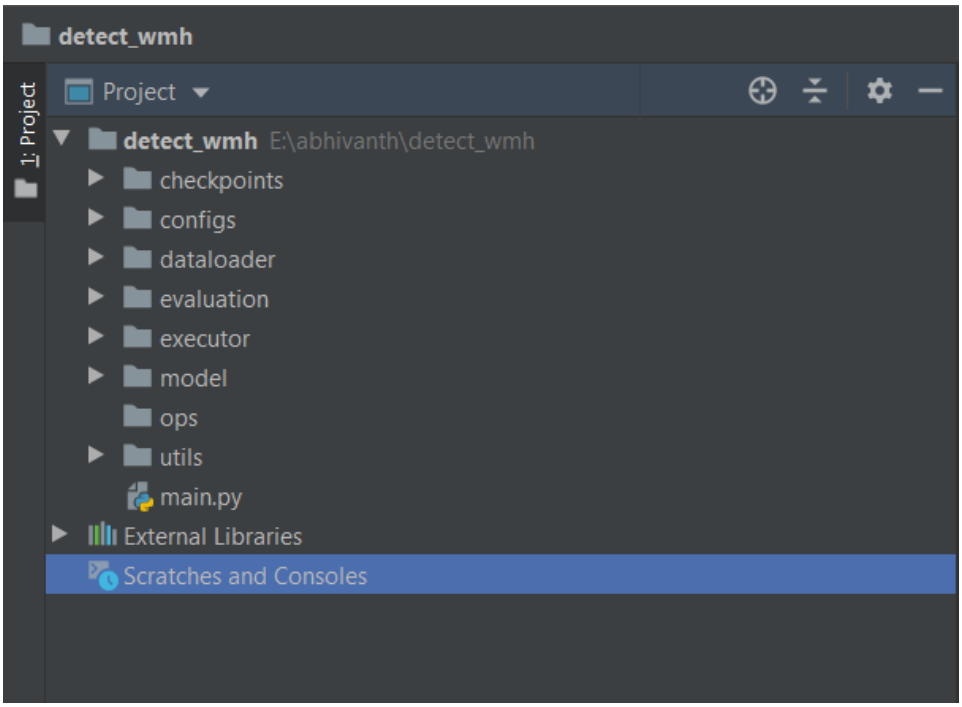

Figure 2: WMH Folder Structure

- *checkpoint* - the checkpoint for each epoch is saved in this directory
- *dataloader* - to preprocess and split the data into training, validation and test data set
- *evaluation* - to test the trained model and calculate the metrics for predicated image
- *executor* - to train the network
- *model* - defines the network model
- *utils* - contains scripts for non-deep learning operations like plotting graphs

On executing main.py, the user is prompted to enter the file location for data set and training case scenario:

- *1 - Held out*
  Now the user is prompted to enter number of images for training
- *2 - Cross validation*
  Now the user is prompted to enter the dataset to be tested
- *3 - Leave one out*

3.3 Evaluation

The 3 different techniques are used to evaluate the performance of the model: Cross-Validation, held-out, and Leave-one-out

3.3.1 Cross-Validation

We split the datasets into 2 parts for training and testing. Out of 3 datasets, UMC Utrecht, NUHS Singapore datasets are used for training, and the VU-Amsterdam ge3t dataset is used for testing. The network is trained for 50 epochs. The line graph is plotted for epoch vs training & validation loss. We calculated the *Dice Similarity Coefficient* and *Recall* for each test subject and finally, the average scores are computed for all test subjects. We have obtained the results with average scores as 0.65 DSC and 0.74 Recall.

3.3.2 Held-Out

In this method, out of 60 subjects, 17 random subjects from each hospital were chosen for training the model and the remaining 9 subjects were used for testing. The network is trained for 100 epochs. The line graph is plotted for epoch vs training & validation loss. We calculated the *Dice Similarity Coefficient*, *Average volume difference*, *Recall*, and *F1-score* for each test subject and finally, the average scores are computed for all test subjects. We have obtained the results with average scores as 0.57 DSC, 57.14 AVD, 0.61 Recall, and 0.54 F1-score

3.3.3 Leave-One-Out

In this method, for every iteration Out of 60 subjects, one subject is used for testing and others for training. Thus, all subjects are used for testing once. The Checkpoint folder is created for every combination of leave-one-out and each folder has checkpoints at each epoch. It returns a text file containing a list of epoch index vs training & validation loss summary for every combination of leave one out. It also returns another text file with metric scores for each subject and also the average score of all subjects.

3.3 Results
3.3.1 Plot Loss Graphs:

The given below are the graphs plotted for training vs validation loss.

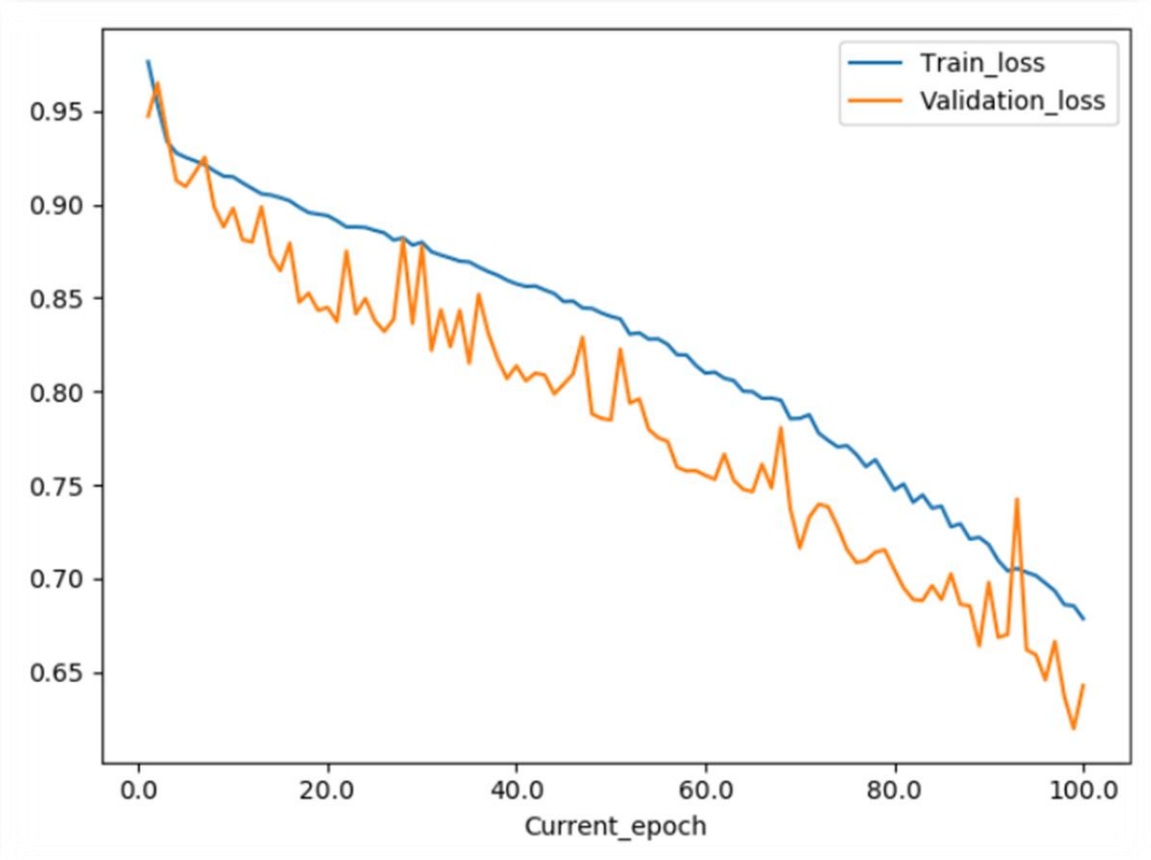

Figure 3: Plot loss Held out Training

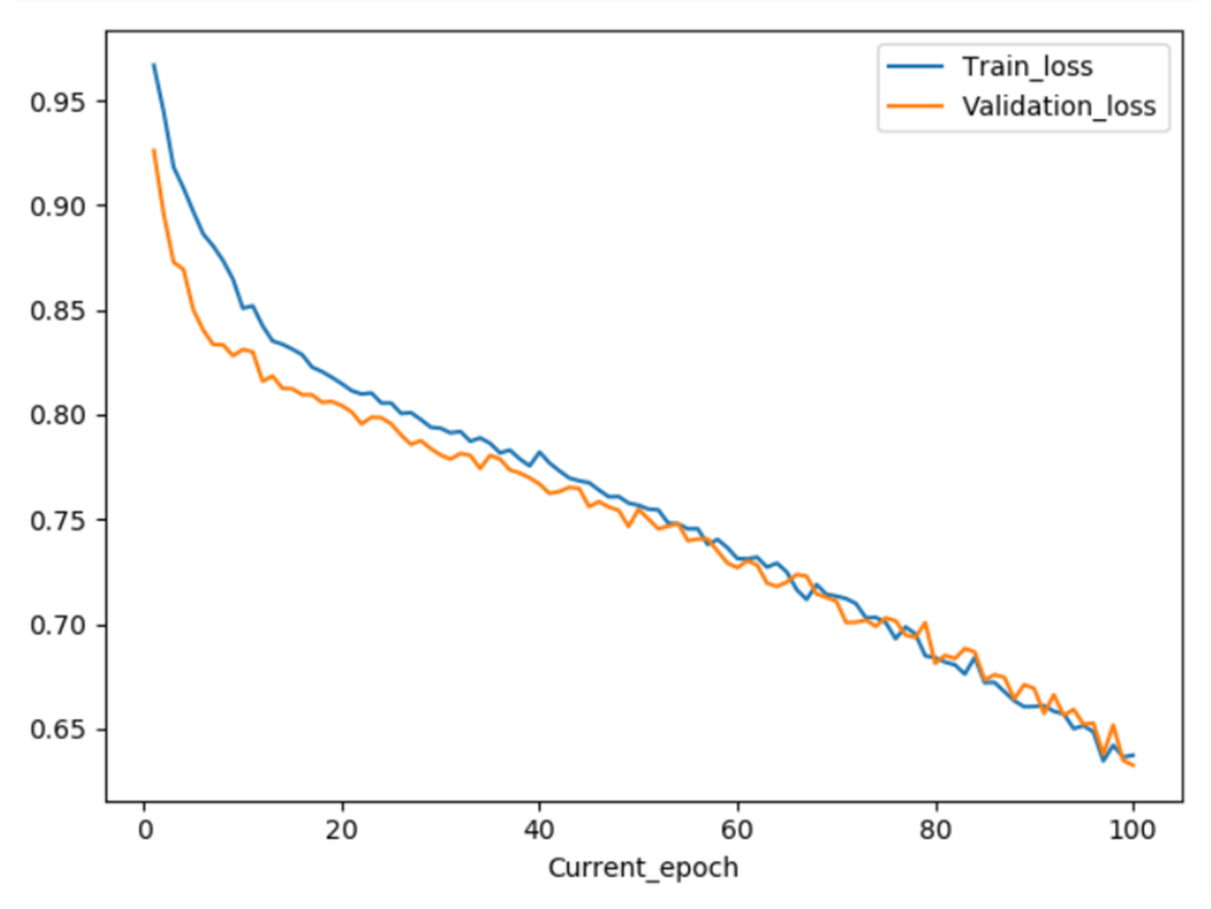

Figure 4: Plot loss cross scanner training

3.3.2 Evaluation Results:

	Average DSc	**Average Avd**	**Average Recall**	**Average f1**
Held-Out-Training	0.574609457299979	57.1745932439558	0.6143315812369892	0.5440149107966883
Cross-Scanner-Training	0.6566297680816892	44.091951540572055	0.7463792908267368	0.6666845678582549

Table 2: Evaluation parameter

3.3.3 Output Comparison and Visualization:

Subject: The subject number 105 is taken into consideration from Amsterdam Ge3t dataset

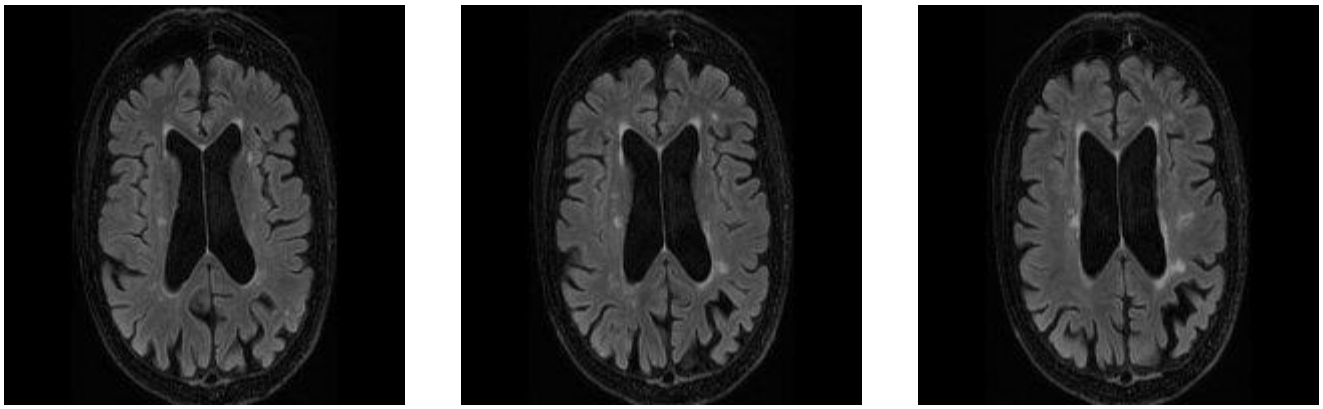

Figure 5: Input slice: (slices 54,55,56; from left to right)

Figure 6: Actual slice: (slices 54,55,56; from left to right)

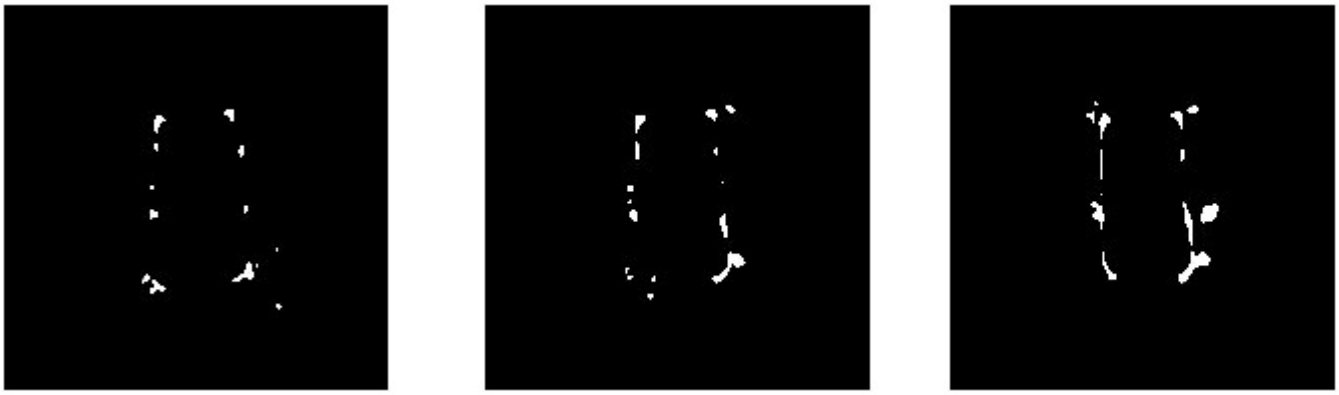

Figure 7: Predicted Slice holdout: (slices 54,55,56; from left to right)

Figure 8: Predicted Slice Cross Scanner testing: (slices 54,55,56; from left to right)

3.3.4 Overlaid View:

The overlaid view of slice 56 of input image (subject 105 of Ge3t data), actual segmentation, predicted segmentation is displayed in *Fig 7* and *Fig 8*. In the image, the red color indicates the actual segmentation and blue color indicates the predicted segmentation

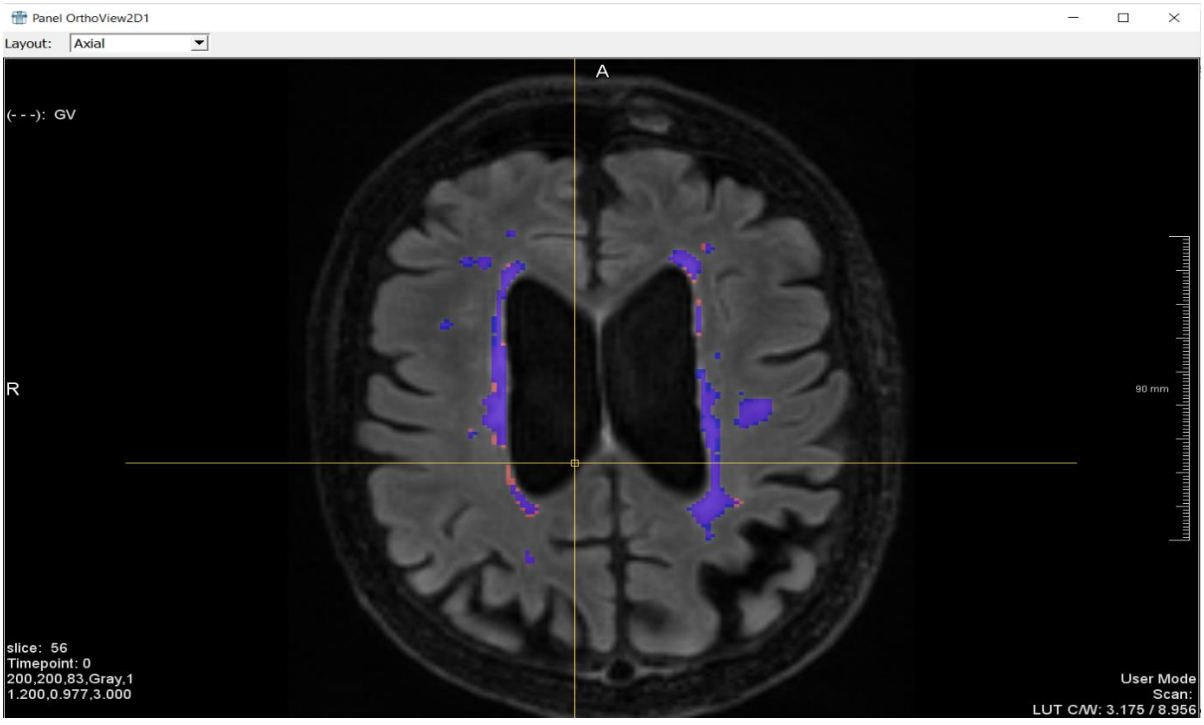

Figure 9: Overlaid View: Held out Training

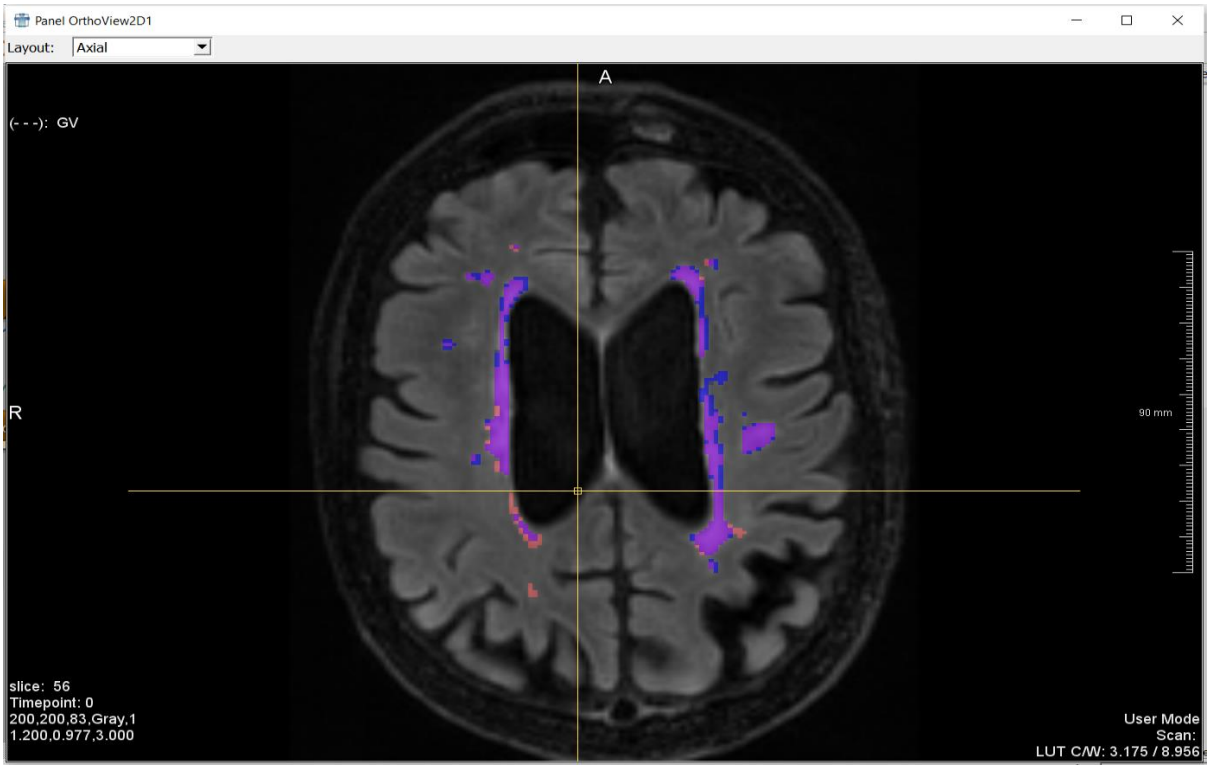

Figure 10: Overlaid View: Cross scanner training

4. CEREBRAL MICROBLEEDS IMPLEMENTATION

4.1 Data preparation & Pre-processing

We considered the ADNI data set of MCH (micro-hemorrhages) type. Initially, we have converted the Dicom images to Nifti images and generated the ground truth images using the coordinates provided in the CSV file. In the next step, the images are normalized in the range of 0 to 1 and the dataset is split with respect to the CMB count for train, test, and validation as per the below table.

	Subjects	Positive Patches
Train	325	2833
Validation	37	571
Test	139	738
Total	501	4142

Table 3: CMB Data Preparation & Pre-processing

Then accordingly the 3d patches are extracted of size 16*16*10. These patches are used for the Screening stage. Finally, we have to append the input and ground truth patches. So before appending the patches, the positive patches of the input image are

augmented. As part of augmentation, the patches are rotated (-12°, +12°), flipped and shifted (-10 & +10 distance in x-axis). For the screening stage, as the dataset has to be balanced, an equal number of negative patches are added randomly as per the positive patches count.

4.2 Architecture

The architecture of the network of screening stage and discrimination stage is explained below.

4.2.1 Screening Stage

We have built the network as per the below table. After each convolutional & fully convolutional layer rectified linear unit (RELU) is used as an activation function and also dropout function is used after every convolutional layer. We have used cross-entropy as a loss function, optimizer as standard gradient descent, and the network is trained with the following hyper-parameters: learning rate = 0.03, momentum = 0.9, dropout rate = 0.3, batch size = 100. The weights were initialized from the Gaussian distribution ($\mu = 0$, $\sigma = 0.01$).

Layer	Kernel Size	Stride	Output Size	Feature Volumes
Input	-	-	16 x 16 x 10	1
C1	5 x 5 x 3	1	12 x 12 x 8	64
M1	2 x 2 x 2	2	6 x 6 x 4	64
C2	3 x 3 x 3	1	4 x 4 x 2	64
C3	3 x 3 x 1	1	2 x 2 x 2	64
FC1	2 x 2 x 2	1	1 x 1 x 1	150
FC2	1 x 1 x 1	1	1 x 1 x 1	2

Table 3: Screening Stage Architecture

C - Convolutional Layer M - Max pooling Layer FC - Fully Convolutional Layer

4.2.2 Discrimination Stage

We have built the network as per the below table. After each convolutional & fully convolutional layer rectified linear unit (RELU) is used as an activation function. We have used cross-entropy as a loss function, optimizer as standard gradient descent, and used the following hyper-parameters: learning rate = 0.03, momentum = 0.9, batch size = 100.

Layer	Kernel Size	Stride	Output Size	Feature Volumes
Input	-	-	20 x 20 x 16	1
C1	7 x 7 x 5	1	14 x 14 x 12	32
M1	2 x 2 x 2	2	7 x 7 x 6	32

C2	5 x 5 x 3	1	3 x 3 x 4	64
FC1	-	-	1 x 1 x 1	500
FC2	-	-	1 x 1 x 1	100
FC3	-	-	1 x 1 x 1	2

Table 4: Discrimination Stage Architecture

4.3 Implementation

The implementation details of screening stage and discrimination stage is explained below.

4.3.1 Screening stage

This stage is composed of 3 sub-steps: i) initially we used a balanced dataset (equal number of positive and negative patches) to train the network, ii) The false positives are extracted from the trained model and the dataset is enlarged which consists of 23.63% positives, 47.52% randomly selected negatives, and 28.85% false positives, iii) then the network is trained with the enlarged dataset to enhance the discrimination capability of the network. After the model is trained, the whole volume of the image is passed to the model which results in a predicted map with 2 channels. In this, the 2nd channel is the probability of CMB in which non-max suppression is applied and then the index mapping to original coordinates is used to get the score map of the possible CMBs. This stage extracts possible candidates to be screened avoiding traversal through the whole image.

4.3.2 Discrimination stage

In this stage, an equal number of positives and false positives determined from the screening stage is used for training. This stage aims at discriminating between the positives and its mimics (false positives).

4.4 Steps to perform the execution

- *checkpoint* - the checkpoint for each epoch is saved in this directory
- *dataloader* - to preprocess and split the data into training, validation and test data set
- *evaluation* - to test the trained model and calculate the metrics for predicated image
- *executor* - to train the network
- *model* - defines the network model
- *utils* - contains scripts for non-deep learning operations like plotting graphs

On executing main.py, the user is prompted to enter weather Dicom to Nifti and to generate ground truth with a yes or no question

- *1 – Create Nifti input and groundtruth images from Dicom files*
  Now the user is prompted to enter Dicom file location, desired file location to save ground truth and Nifti images and metadata file (csv file containing the ground truth coordinates)
- *2 – Execute screening stage 1*
  Now the user is prompted to enter the file location containing Nifti files created in the previous step.
- *3 – Execute screening stage 2*

Now the user is prompted to enter the file location containing Nifti files created in step 1 and location of desired checkpoint from screening stage 1

- *4 – Get score map*

  Now the user is prompted to enter screening stage 2 checkpoint and test file location

5. References

1. Fully Convolutional Network Ensembles for White Matter Hyperintensities Segmentation in MR Images *(Hongwei Li et al)*
2. A post-processing method to improve the white matter hyperintensity segmentation accuracy for randomly-initialized U-net *(Yeu Zhang et al)*
3. Fully Convolutional Networks for Semantic Segmentation *(Jonathan Long et al)*
4. U-Net: Convolutional Networks for Biomedical Image Segmentation *(Olaf Ranneberger et al)*
5. Automatic Detection of Cerebral Microbleeds From MR Images via 3D Convolutional Neural Networks *(Qi Dou et al)*
6. Cerebral Microbleeds Detection via Convolutional Neural Network with and Without Batch Normalization *(Jin Hong et al)*
7. H. J. Kuijf *et al*., "Standardized Assessment of Automatic Segmentation of White Matter Hyperintensities and Results of the WMH Segmentation Challenge," in *IEEE Transactions on Medical Imaging*, vol. 38, no. 11, pp. 2556-2568, Nov. 2019, doi: 10.1109/TMI.2019.2905770.