CS512 - Final Project Report

Automating segmentations of sub-cortical labels using deep learning on Quantitative Susceptibility Mapping

Group 1:

- Rasheed Abid, A20500507, rabid@hawk.iit.edu
- Khalid Saifullah, A20423546, ksaifullah@hawk.iit.edu

Department of Biomedical Engineering
Illinois Institute of Technology
Submission date: 04/19/2023

Associated data/files/scripts can be found here (with HawkID logged in):
https://drive.google.com/drive/folders/1kuUpMAcF4c77dDfCE6Ot1C9P_jXDM7FK?usp=sharing

Main paper: Chai, C., Qiao, P., Zhao, B., Wang, H., Liu, G., Wu, H., Haacke, E.M., Shen, W., Cao, C., Ye, X. and Liu, Z., 2020. Automated Segmentation of Brain Gray Matter Nuclei on Quantitative Susceptibility Mapping Using Deep Convolutional Neural Network. *arXiv preprint arXiv:2008.00901*

---

1. Introduction and problem statement

Quantitative Susceptibility Mapping (QSM) is an advanced magnetic resonance imaging (MRI) technique that can measure magnetic susceptibility variations within tissues. This technique has shown great potential in the diagnosis of various neurological disorders, such as Parkinson's disease and Alzheimer's disease. However, working with QSM data presents several challenges, including background-foreground masking, very low contrast, and MRI artifacts such as ringing and checkerboard. Therefore, automating the 3D segmentation of masks or labels, which is a very tedious task if done by hand, is crucial yet challenging. To get an edge over the noise and problems of the modality, many researchers focus to get information from the different modalities of data [1][3] as well.

The objective of this project is to implement a 3D segmentation model on Quantitative Susceptibility Mapping (QSM) data ***only***, using a deep learning approach, as an implementation and upgradation of the proposed research in [1]. QSM data has noise, resolution, and contrast issues that require preprocessing before feeding into a neural network. The proposed method will be mostly based on a previously published research paper, but we will use a larger and higher resolution dataset, which brings additional challenges of memory and computational power.

2. Dataset

A dataset of 109 subjects with 5 sub-cortical labels in common were collected for this project. The dataset is a subset of the study [2] and as the dataset is not published for the public yet, we are only able to provide 5 random subjects with their identity removed. The provided subjects will have their original QSM

in their native space, transformed QSM in their atlas space (step 1 of the preprocessing), 5 available label masks (step 2(c) of the preprocessing) and their downsampled versions (step 2(b) of preprocessing). These should be enough to visualize and test the models. A demo will be provided in the scripts.

3. Methods

In this project, we compared two 3D CNN-based methods to segment gray matter nuclei from the QSM. The two models are 3D-UNet and 3D-ResUNet. Experimental results from the paper implies that the QSM had more contribution in distinguishing the subcortical structures, which made it possible for accurate quantitative susceptibility assessment when 3D T1WI was absent. The QSM, despite its low structural resolution, has the advantage in presenting the iron-accumulating deep gray matter nuclei.

In the domain of medical image segmentation, the U-Net architecture and its variations have demonstrated remarkable accuracy in 2D image segmentation tasks, such as identifying cell nuclei or cell boundaries in histopathological images. However, when dealing with 3D images like CT and MR images, simply treating the 2D slices as separate images and applying the same methods used for 2D segmentation would result in lower segmentation accuracy. This is because the CNN model is unable to identify the relationship between the adjacent slices, which leads to the loss of inter-slice contextual information.

In deep learning, the GPU has to store not only the input images, but also the activation maps of all neurons to compute the gradients during training. Therefore, when dealing with 3D volumetric images, one usually faces a pressing dilemma between memory efficiency and segmentation accuracy. Also, the segmentation accuracy will be lost due to the absence of spatial contextual information if we treat the volumetric image slices as separate 2D images, while the training may become infeasible if we feed the whole volume into the network. One of the solutions to the dilemma is to split the whole volumetric image into 3D image patches, which is considered to be beneficial in reducing memory cost while preserving the inter-slice contextual information. However, it in turn brings another spatial contextual information loss, as the fields of view (FoVs) of the CNNs were strictly limited by the patch size.

Despite the fact that a deeper network has been shown to be more expressive than its shallower counterpart, simply stacking many convolution layers would impose significant difficulties in training due to the so-called gradient-vanishing problem. This can be solved by introducing residual structures to the CNN. ResNet is able to build very deep networks, and achieved remarkable performance in image recognition tasks. With residual structures, the networks have no spurious local optima, making it easier to converge.

4. Approach

4.1 Preprocessing

Step 1: As mentioned, the dataset is in their native space, which is not an ideal case scenario for labeling. To bring all the native space data to a predefined template (known as the brain atlas), we use the ANTS registration [5] toolbox, an industry-wide used toolbox for nifti image registration. Though, this process is beyond the scope of this project, as this is a must step, we will provide a demo script that was used to bring the brain images from native to the atlas space. We will be using the [2] atlas space for our project. To elaborate about this portion we did the following sub-steps:

a. Background Removal: The image background is removed to focus on the relevant brain structure.

b. Image Normalization: The image is normalized to the atlas scale, ensuring uniformity across the dataset. It is then masked again to achieve consistent data representation. This is done, because QSM data doesn't have a background of zero. The background of QSM data is dependent on the CSF values of the patient at the time of reading.

c. Parameter selection: Based on atlases and modalities, we had to select the suitable parameters for the registrations.

d. Registration: The QSM images are registered to the MIITRA atlas using the ANTS registration toolbox.

Step 2: Once registered to the atlas, the following steps were done, with each step being checked rigorously to ensure the expected data (this is just a precaution step, as nifti images with their header information can easily get patched up).

a. Padding: The original images, once transformed to the atlas space, are of the dimension 380x440x380 with high resolution of 0.5x0.5x0.5mm^3 per voxel. We padded the image with zeros to make it a cubicle of 512x512x512 images for our use case.
b. Downsample: We do not have the GPU and memory resources to train our model that can accommodate 512x512x512 size images. So, we needed to downsample the images without losing too much information. We downsampled the images and their corresponding masks by a factor of 2 on each axis to generate 3 versions: 256x256x256, 128x128x128 and 64x64x64 images. In the case of masks, it was binarized to avoid problems further down the road.
c. Label preprocessing: The initial labels were provided in the form of FreeSurfer label lists and a consolidated Nifti image containing masks for both left and right hemispheres. Our task involved splitting these masks into separate label masks, merging the left and right hemisphere masks for the five common labels, downsampling the labels as per step 2(b), and converting them into binary if necessary.

4.2 Designing architecture and tuning hyper-parameter

U-Net has been one of the most successful structures in medical image segmentation. It is designed as a symmetric encoder-decoder structure, with several skip connections between the encoder and decoder to refine the segmentation results. Our first experimental model is the 3D U-Net.
Second, the stacked convolutional layers in the original U-Net are replaced by a residual block as shown in the figure below. As mentioned previously, the residual structure enables the network parameters to be updated from the beginning, and elegantly solves the gradient vanishing problem. This is our second experimental model - 3D ResUNet.

Due to limited GPU memory, the images are divided into patches of size 64x64x64 before being fed into the network. Note that although feeding patches instead of the whole image has been a regular approach in 3D CNNs, only utilizing local context information from the patches may lead to segmentation performance loss.

UNet and ResUNet are two popular deep learning architectures commonly used for semantic segmentation tasks, such as image segmentation.

The UNet architecture was introduced in 2015 by Ronneberger et al. It consists of an encoder and a decoder network, which are connected by a bottleneck layer. The encoder network is used to capture the contextual information of the input image by applying several convolutional and pooling layers. The

decoder network then reconstructs the output image by applying several up-convolutional and concatenation layers, which help to preserve the spatial information lost during the pooling operation in the encoder network. The UNet architecture has been widely used in medical image segmentation tasks.

The ResUNet architecture is an extension of the UNet architecture that was proposed in 2018 by Jha et al. It incorporates residual connections between the encoder and decoder networks to help alleviate the problem of vanishing gradients. The residual connections are used to propagate information from the encoder to the decoder network, thereby helping to improve the accuracy of the segmentation. The ResUNet architecture has been shown to outperform the original UNet architecture on several benchmark datasets.

Both UNet and ResUNet architectures have been widely used in the field of computer vision and have achieved state-of-the-art performance on several benchmark datasets.

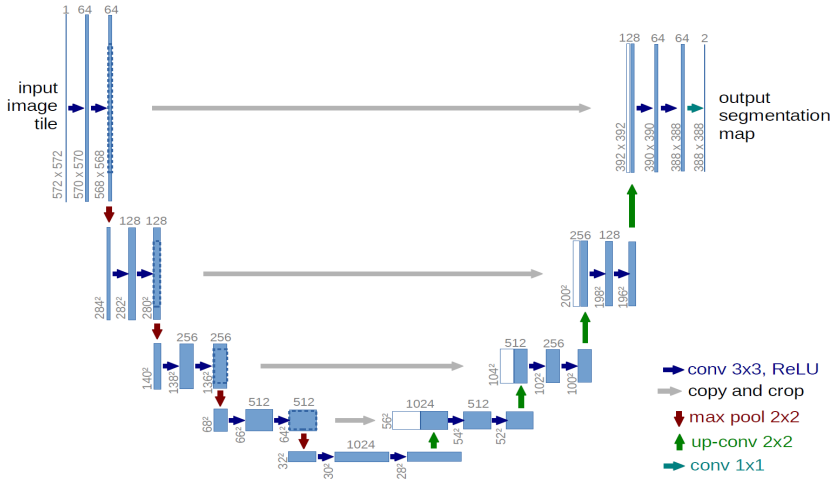
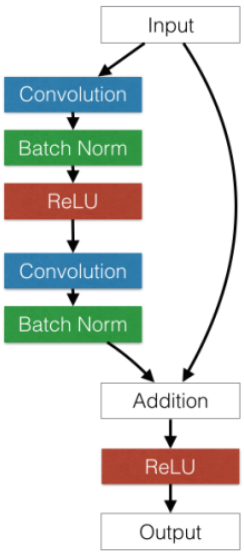


Figure 1: Architecture of UNet



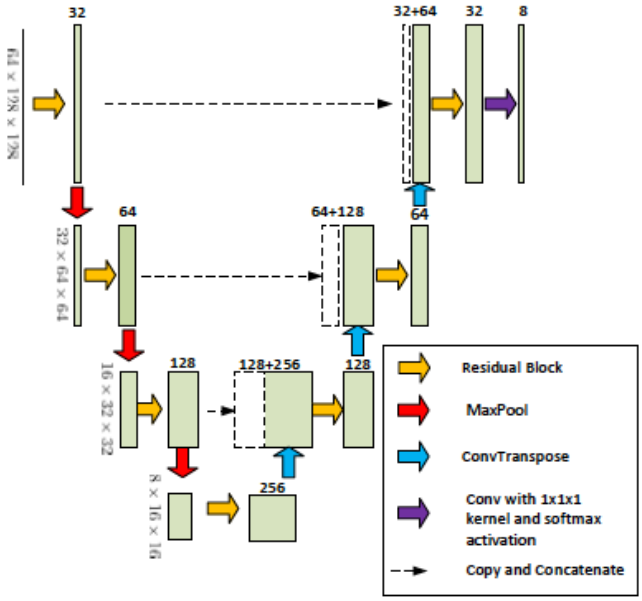


Figure 2: Architecture of Residual block

Figure 3: Architecture of ResUnet

We adopted Adam method as the optimizer, and set the initial learning rate as 3x10^-4 with β1=0.9 and β2 = 0:99, with a weight decay of 3x10^-5. Due to limited memory space on a single GPU, the batch size was set to be 4.

5. Evaluation, System and results

5.1 Evaluation metrics

For evaluation we are using Dice similarity coefficient, a standard surface matching parameter, as it is used in the paper [1]. Furthermore, we used Hausdorff distance to measure the dissimilarity of the ground truth and the predicted masks.

Dice similarity:

The Dice similarity coefficient, also known as the Sørensen–Dice index, is a measure of similarity between two sets of data. It is commonly used in image segmentation and pattern recognition. Given two sets A and B, the Dice similarity coefficient is defined as:

$$\mathbf{Dice(A, B) = 2 * |A \cap B| / (|A| + |B|)}$$

where |A| and |B| represent the cardinality of sets A and B, and |A ∩ B| represents the number of elements in the intersection of A and B. In our case, the set A and B represents the ground truth mask and the predicted mask for labels.

Hausdorff distance:

The Hausdorff distance is a mathematical measure used to quantify the dissimilarity between two sets of points in a metric space. It calculates the maximum distance between any point in one set and its nearest point in the other set. Formally, let A and B be two subsets of a metric space M, then the Hausdorff distance H(A, B) between A and B is defined as:

$$\mathbf{H(A, B) = max\{sup\{dist(a, B)\}, sup\{dist(b, A)\}\}}$$

where a belongs to A, b belongs to B, and dist(x, Y) denotes the distance between the point x and the set Y. In our case, A and B are the ground truth mask and predicted mask respectively.

5.2 System

We have tried and tested a lot of possibilities for our architecture. Though all of the runs require high computational power, parallel running helped us test our possibilities broadly. We tested our program on google colab pro (with 25GB CPU memory and 15GB GPU memory). For other tests, such as transformations, we used a local computer (macOS, 3.2GHz Quad-code intel core i5, 16GB memory).

5.3 Results

A typical training for *hippocampus* has the following train-validation loss curves (Figure 4) and dice similarity curve(Figure 5).

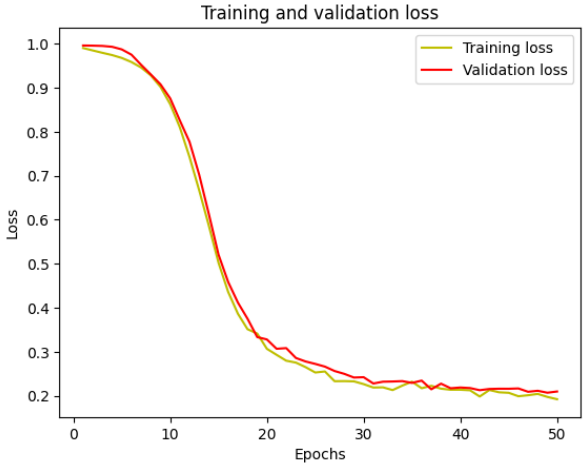


Figure 4: The loss curve for Hippocampus

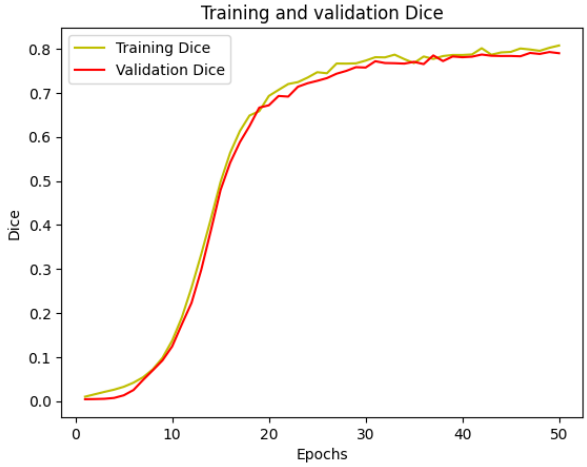


Figure 5: The dice similarity curve for Hippocampus

To check what the learning algorithm is really visualizing, we checked individual patches after learning and predicting, and we plotted some slices of the patches (Figure 6). Here is an example of a slice of a patch from *putamen.* To remind you, putamen is about 0.5-1.0% of the brain volume.

Original brain image Ground truth Predicted image

Figure 6: Visualizing a set of patches, where the original brain potion, a portion of ground label of putamen and our predicted patch by the model is shown.

To understand the overall resulting matrices of dice and Hausdoff values, we train-validated-tested all the labels. The values for the following parameters are provided below.

Model: UNet
Number of train-validation subjects: 56-24 (total 80)
Number of test subjects: 20
Number of epochs: 50
Image dimension: 128x128x128
Table 1 shows training and testing results for the brain mask and the selected labels for our model for the parameters mentioned above.

Serial	Name of the mask	Validation DS	Testing DS	Testing HD
1	Whole brain mask	0.99	0.97	0.04
2	Hippocampus	0.79	0.82	1.24
3	Putamen	0.81	0.79	1.48
4	Thalamus	0.87	0.87	1.46
5	Amygdala	0.60	0.50	0.94
6	Caudate	0.76	0.81	1.43

Table 1: Average validation dice similarity, average testing dice similarity and average Hausdorff distance (in voxels) for the brain masks and available five labels with 80 train subjects and 20 test subjects, with 50 epochs of learning on 128x128x128 images on UNet architecture.

Comparing the results with the paper [1]:

Input	CN	GP	PUT	THA	SN	RN	DN
QSM	0.782	0.828	0.819	0.827	0.690	0.680	0.714
T_1WI	0.760	0.778	0.799	0.825	0.589	0.635	0.338
QSM +T_1WI	0.802	0.840	0.827	0.844	0.719	0.763	0.774

Table 2: Average Dice similarity coefficient in the paper [1] for caudate, putamen and thalamus with QSM data *only*. Notice that our model performs better on caudate and thalamus.

Table 2 shows the results reported in the paper. Here, they mentioned 8 labels, and tested on mulitmodalities. For the QSM only (highlighted), we had 3 subjects in common, the caudate, putamen and thalamus, and our models performed better in caudate and thalamus. The difference, however, is that they used Double Branch ResUNet.

To check how learning of the model consisted of downsampling, we tested a few number of cases. Though, as we considered higher dimensions, this quickly exhausted the system. Thus we had to change variables, such as number of subjects, number of epochs etc. We present the results for *hippocampus, caudate* below in table 3.

Label name	Dimension	Epochs	Dice Similarity			
			UNet		ResUNet	
			No. of Subjects	Dice Similarity	No. of Subjects	Dice Similarity
Hippocampus	256x256x256	50	100	0.81	41	0.68
Hippocampus	128x128x128	50	100	0.80	100	0.71
Caudate	256x256x256	100	100	0.65	100	0.64
Caudate	128x128x128	100	100	0.76	100	0.76

Table 3: Comparison of dice similarity results between Unet and ResUNet.

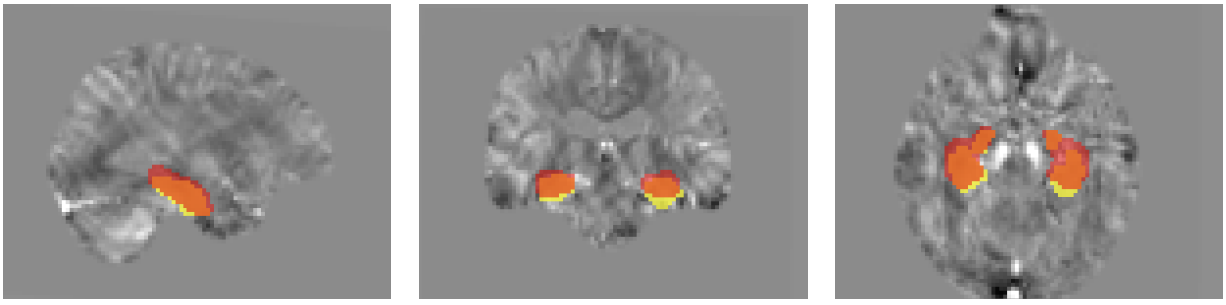
ResUNet converges faster but has a greater number of trainable parameters which is around 23 million in contrast to the 5.5 million of UNet. It can be seen from table 3 that caudate with 128x128x128 dimension has almost similar performance between UNet and ResUNet. However, with higher resolution i.e. 256x256x256, the model's performance decreases from approx. 0.76 to 0.64. On the other hand, the results on resolution performance are not conclusive for the hippocampus as the number of subjects between 128x128x128 and 256x256x256 were different. However, UNet achieves better performance than ResUNet with 128x128x128 for hippocampus.

The reason for the lesser number of subjects for 256x256x256 image shape was because of the limitation of memory and gpu resources on google colab. Higher resolution data is more computationally expensive to process and requires more resources such as memory and processing power.

Increasing the resolution of the input data may not always improve the performance of the model. In such cases, it may be better to use data augmentation techniques to artificially increase the amount of training data and improve the model's robustness to variations in the input data.

From table 2 and 3 combined, we may claim that both our UNet and ResUNet give 0.76 dice similarity as opposed to 0.78 dice similarity in the paper with double branch ResUNet. Having said that, we didn't implement data augmentation as we were limited by computational resources. Thus, our results without additional data were close to the results shown in the paper and could possibly be equal to their performance with data augmentation and given resources.

After all the training, validation and testing; once we reconstruct the images from the set of 64x64x64 patches to its original 128x128x128 Nifti form, the result looks like this in a typical *Hippocampus* presented on the whole brain. **Red denotes the predicted value, whereas yellow is the ground truth.**



Sagittal view Frontal view Axial view

Figure 7: Brain image with its predicted (red) and ground truth (yellow) mask, after reconstruction is done from the patches to the whole brain.

5. Hurdles

Atlas conversion: Though it has been proposed that we will work our project on three different atlases, due to time and computational power limitation we were only able to work on the highest resolution atlas.
Conversion loss: To get the raw QSM data to fit into our model, we need to pass through many steps (section 3) which definitely attributes to the loss of information in our case. So, even though we had high resolution images, with all the preprocessing, the loss increases.

Preprocessing: QSM data requires preprocessing to reduce noise while preserving the relevant information. This requires expertise in image processing techniques [1]. However, with the labels provided, we only needed to preprocess the histogram according to the needs.

High-Resolution Data: The higher resolution dataset will require more computational power and memory for the training process. Therefore, the implementation needs to be optimized to handle the larger dataset. We carefully downsampled the data, and checked multiple architectures to better fit our cause.

Different modalities issues: As we didn't have T1 weighted imaging data, we only reported the scores based on QSM images.

Small volumetric labels: For the brain labels we used for the project, we had volumes that are significantly smaller in dimension compared to the whole brain. This led us to selecting more subjects with longer epochs to achieve good results.

Neural Network Design: The neural network needs to be designed carefully, considering the size of the input data, the number of layers, and the type of activation functions to be used.
Hyperparameter Tuning: Finding the optimal hyperparameters for the model is crucial to achieve high accuracy in segmentation. This requires a thorough understanding of the model architecture and data properties.

6. Future works

To tackle the loss of FoVs that comes from splitting 3D volumetric images into patches, we would like to adopt a double branch network that takes patches with both original and low resolution as input. Memory and computation is a big problem with 3D image segmentation, especially for QSM. Further research on optimizing the results or toning down the preprocessing loss is expected.

7. Conclusion

This project aimed to implement a 3D segmentation model on QSM data using deep learning, based on a previously published paper. The challenges of larger and higher resolution data required careful preprocessing, neural network designing, and hyperparameter tuning to achieve higher accuracy in segmentation, for small volumetric label data. The results of this project will have significant implications for the diagnosis of neurological disorders, if researched further.

8. Walkthrough over the project files, scripts

8.1 Directories

The codes are generated in such a way that under a “main_dir”, the following folders will be contained. This convention must be followed to ensure the smooth running of the project.

- data_dir: Contains the data. Subdirectories are:
    - {subject_id}: contains the data files. Naming convention is mentioned in (section 8.2)
- files: Contains the files that are used for training and testing.
- results_dir:
    - reconstructed_predictions: contains nifti files, which are reconstructed from the predicted patches after we run the script *testing_pipeline.ipynb*
    - saved_models: Here we have the saved models from our training phase. Follow section 8.2 to understand naming convention.
    - saved_notebooks: We saved some of our python notebooks with the outputs to check and learn about the results. Some may have errors in it, but all of them contain training information. We don’t guarantee that all of these will run smoothly, but during our train-testing phases, we had these outputs, which shaped our project. These are just a reference so that you can have a look and check our progress.
- scripts_dir: contains finalized scripts for the project.
    - draft_scripts: this folder contains all other scripts from bash and some of our tuning and testing cases. As our project heavily relied on macOS and bash due to Nift files, FSL and ANTS, we are attaching them to understand how depth we had to work out the alignment problems for brain images. {rabid_runs} contains all the major scripts that was used for the project by Rasheed.
- report_writing: contains the report, presentation and other results related images and figures.
- temp_dir: contains some temporary files.

8.2 Naming convention:

Here is a brief description of the important files and their naming convention that need to be looked out for:

- filename_subXXX.txt: contains XXX number of names of the subject (or their subject ids) of that will be required for training and validation. (for the sake of confidentiality of the patients, we are replacing the numeers with 1, 2, 3 and so on.)
- filename_testingXXX.txt: contains XXX number of names of the subject (or their subject ids) of that will be required for testing. As the general convention, this data is unseen to the model during its training. (for the sake of confidentiality of the patients, we are replacing the numeers with 1, 2, 3 and so on.)
- QSM_dimXXX_w_bg.nii.gz: Original QSM data, downsampled to reduce memory to the dimension XXX with steps of 2. (So, if the file is QSM_dim256_w_bg.nii.gz) , it means the data is of the dimension 256x256x256, which is downsampled by 2 from 512x512x512, and if the data file is QSM_dim128_w_bg.nii.gz, it means it is further downsampled by 2, from 256x256x256).
- QSM_brain_mask_dimXXX.nii.gz: Same convention for the corresponding original image, but this is the file of the brain mask.
- QSM_masked_dimXXX.nii.gz: Same convention for the corresponding original image, but this is the file of the brain masked out, before any kind of normalization is performed. To remind, after normalization, background don't stay zero anymore. You need to mask again, if you need to change.
- "label"_mask_down_by_X.nii.gz: The label mask file for the corresponding original file. Available labels for this project are = ["amygdala", "caudate", "hippocampus", "putamen", "thalamus"]. The original labels from the collected dataset had labels that were left and right splitted. We masked them, and merged them to be in the same mask file.
- "label"_mask_dimXXX_subYY_eZZZ.h5: Models saved in the **saved_models** directory after running the training and validation phase. Here the available labels are ["brain" (referring to the brain mask, "amygdala", "caudate", "hippocampus", "putamen", "thalamus"]. XXX refers to the dimension of the image [512, 256, 126], YY refers to the number of subject it was trained on and ZZZ refers to the number of epochs.

8.3 Walkthrough the codes

Before we proceed with our project, we highly recommend that you install either **FSLeyes** (most recommended) or ITKSnap on your local computer to view the Nifti files and visualize. Both of these are pretty lightweight and easy to install with short installation guidelines here:

Installing FSLeyes: Installing FSL (Amherst Workshop Prep)
Using FSLeyes: Using FSLeyes, Part 1: Display and View Functions
Installing ITKSnap: 1A Download and Introduce ITK-SNAP

As the main dataset is still not publicly available, we are attaching the training of a label on the QSM, with the notebook outputs for you to see and visualize the results. As there are steps that require usage of GPUs or other applications, we are breaking down the entire pipeline into chunks for simplicity. There are four basic sections for this -

1. Reading dimensions and padding to satisfy the needs of our structures.
   Script name: **nifti_padding_with_python.ipynb**
2. Downsampling Nifti with correct headers and alignment.
   Script name: **nifti_downsampling_with_python.ipynb**
3. Final training pipeline. In this script, we have generalized the variables for easier use, we have implemented the models and tested and tuned it according to our needs, saved them in a directory for further testing. As we the whole dataset isn't available for training again, we are

providing with a notebook with its outputs in their cells.
Script name: **label_training_pipeline.ipynb**

4. For testing purposes, we are providing a set of data as mentioned in 8.1. This pipeline can be used for re-running and testing our project. The script has ample comments to follow as a guide, but to brief, you only need to change a few variables in cell number **6** and **9**. To get the correct model, one must change the name of the labels and with preloaded variable names, the testing pipeline should run itself. To change echo values, or other parameters, crosscheck if the model is available in the directory **saved_models** and select parameters accordingly the naming convention.
   Script name: **label_testing_pipeline.ipynb**
5. We also had many bash scripts and other train-testing jupyter notebook running for our many steps. As we had to submit the final python scripts, we are attaching them in the directory **draft_scripts** to further emphasize the depth of our work.

9. Team members' responsibility:

(We acknowledge that we didn't have hands on to the authors code of the original paper or any other known implementations of the paper.)

Rasheed Abid	Data collection, image registration, label processing, image preprocessing, prediction reconstruction, and evaluations, partially helping with the architecture, portion of reports and presentation, including all the common working sections.
Khalid Saifullah	Implementing unet, res-unet, architecture modification, hyper-parameter tuning, partially helping with evaluation, portion of report and presentation.

10. References

[1] Chai, C., Qiao, P., Zhao, B., Wang, H., Liu, G., Wu, H., Haacke, E.M., Shen, W., Cao, C., Ye, X. and Liu, Z., 2020. Automated Segmentation of Brain Gray Matter Nuclei on Quantitative Susceptibility Mapping Using Deep Convolutional Neural Network. *arXiv preprint arXiv:2008.00901*.

[2] Niaz, M.R., Wu, Y., Ridwan, A.R., Qi, X., Bennett, D.A. and Arfanakis, K., 2021. Development and evaluation of high‐resolution gray matter labels for the MIITRA atlas. *Alzheimer's & Dementia*, *17*, p.e052575.

[3] Yu, B., Li, L., Guan, X., Xu, X., Liu, X., Yang, Q., Wei, H., Zuo, C. and Zhang, Y., 2021. HybraPD atlas: Towards precise subcortical nuclei segmentation using multimodality medical images in patients with Parkinson disease. *Human brain mapping*, *42*(13), pp.4399-4421.

[4] Jung, W., Bollmann, S. and Lee, J., 2022. Overview of quantitative susceptibility mapping using deep learning: Current status, challenges and opportunities. *NMR in Biomedicine*, *35*(4), p.e4292.

[5] Avants, B.B., Tustison, N. and Song, G., 2009. Advanced normalization tools (ANTS). *Insight j*, *2*(365), pp.1-35.

[6] Ronneberger, O., Fischer, P. and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18* (pp. 234-241). Springer International Publishing.

[7] Jha, D., Smedsrud, P.H., Riegler, M.A., Johansen, D., De Lange, T., Halvorsen, P. and Johansen, H.D., 2019, December. Resunet++: An advanced architecture for medical image segmentation. In *2019 IEEE International Symposium on Multimedia (ISM)* (pp. 225-2255). IEEE

[8] We took pointers and help from the scripts found here: https://github.com/bnsreenu/python_for_microscopists