Project Proposal

Predicting hemorrhage related outcomes with CT volumetry for traumatic hemothorax

Benjamin Albert Chang Yan Gary Yang

Table of contents:

1. Clinical Motivation

Hemothorax refers to the condition where blood presents in the chest. Each year, more than 300,000 cases of hemothorax are identified [1]. It is often diagnosed using radiography, ultrasound, or computed tomography (CT). Among the three imaging modalities, CT is the most sensitive at detecting smaller scale hemothorax [2], and it allows radiologists rather accurately to quantify blood volume, though manual segmentation is required. Drawing contours by hand is a time-prohibitive procedure in an emergency, and therefore our work can potentially better assist surgeons with operation planning.

We envision our work producing an algorithm where it will automatically highlight masks indicating hemothorax for CT images. We hope to deliver a GUI-program to ease visualization eventually, which will also show our segmentation's confidence level.

2. Prior Work

Prior models that studied pleural effusion (excess liquid in general, including blood and water), a condition comparable to hemothorax, are generally rule-based or atlas-based [3]. Therefore they are insufficient at handling anatomical distortions, including the heterogeneity of attenuation and traumatic lung problems.

U-net, as the most fundamental convolutional network for semantic segmentation, remains the most popular base architecture. It is readily used for medical image analysis. However, due to the multifocal nature of hemothorax, analogous to hemoperitoneum [4], convolutional networks will unlikely to perform well.

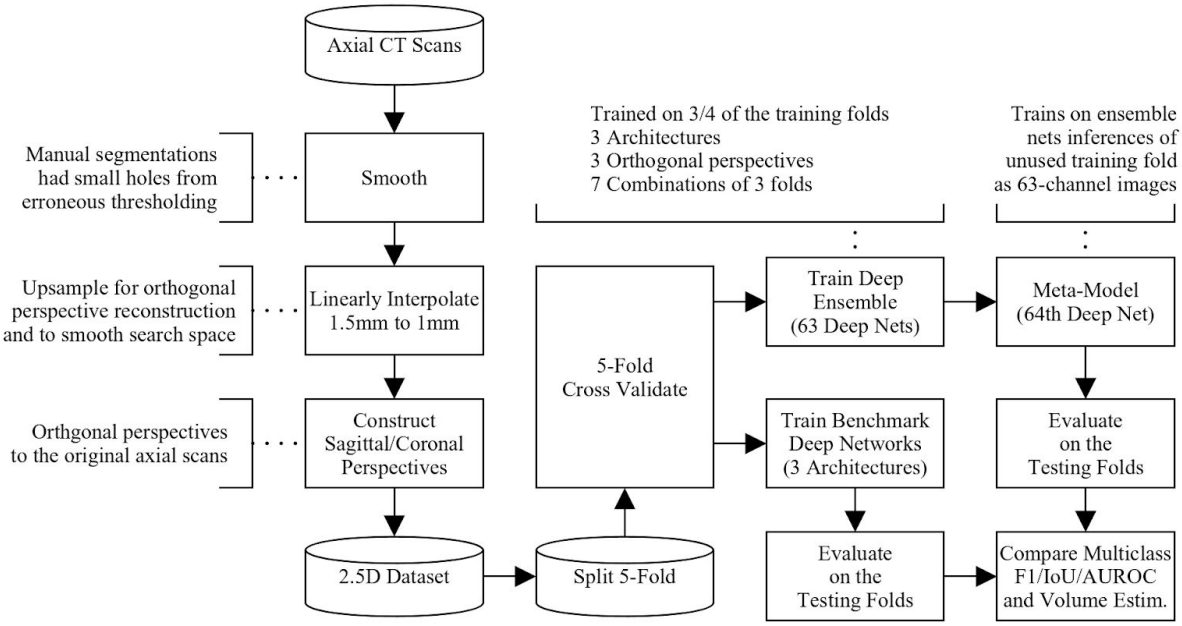
3. Goals

Minimally, we produce a functioning algorithm that inputs a set of CT scans and computes the blood volume, if any. We will use k folds validation to predict this volume, and we expect the error between our prediction and manual-labeled accurate volume estimation to be less than 5%. If time permits, we intend our program to express the prediction's certainty level or interpretable for doctors to assist with surgical decision-making.

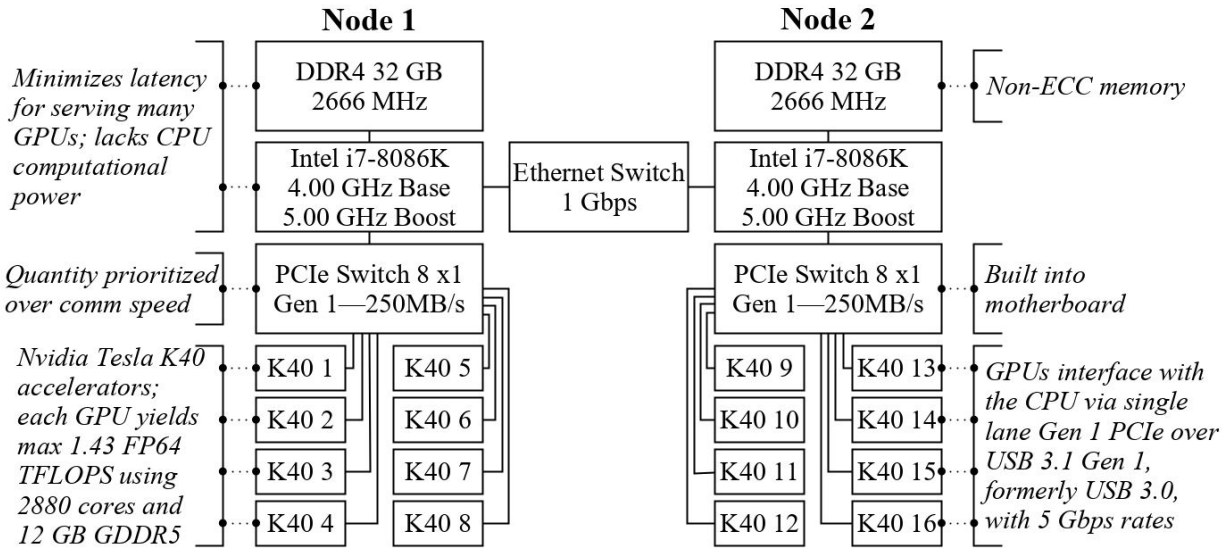
4. Technical Approach

The dataset consists of axial CT scans with 1.5mm voxel resolution from 94 patients. In total, the dataset is approximately 90 GB before preprocessing. Preprocessing involves

three primary stages; smoothing, interpolation, and construction of sagittal and coronal perspectives. Smoothing is necessary to fill holes that are erroneously present from noisy manual labelling; it is applied only to the segmentation masks. Trilinear interpolation is used to generate 1mm voxels so that additional sagittal/coronal slices can be generated. This is useful for network training as it smooths the objective functions. However, in total, the preprocessing multiplies the dataset size in memory by 4.5 fold, reaching approximately 400 GB, averaging 4 GB per patient.



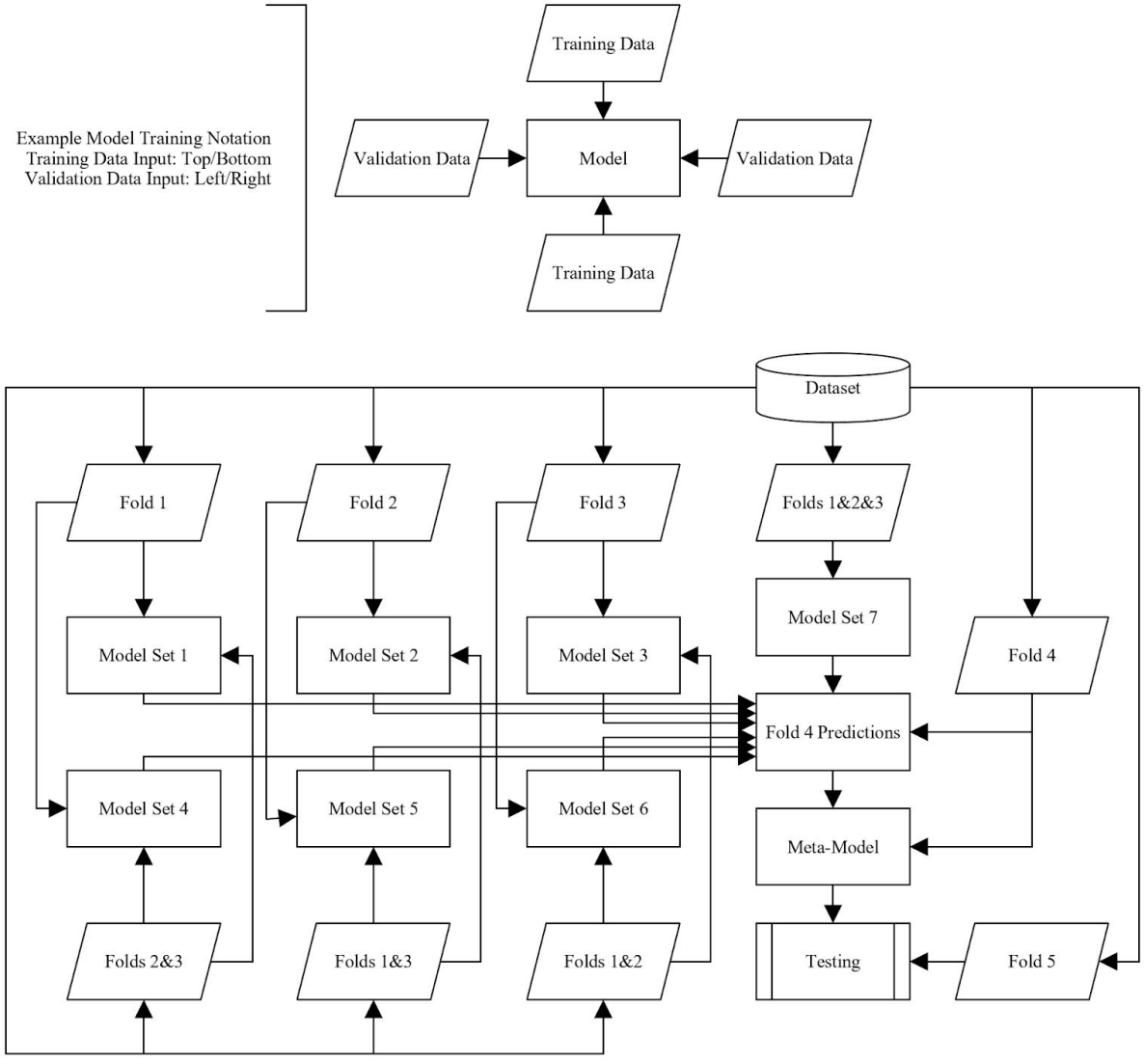
After preprocessing, the data is split for 5-fold cross validation. Then, two diverging paths are implemented: first, the standard approach of applying deep segmentation networks using train/test and train/val/test splits, and second, an ensemble of the deep segmentation networks trained on combinations of folds, the results of which are merged using a meta-model. The meta-model is also a deep network that receives 3D input where each channel is the segmentation mask of one of the ensemble networks. Given this ensemble architecture, a total of 5(21Arch+1) deep networks need to be trained. In this particular implementation of the ensemble architecture, Arch=3: 3D U-Net [7], V-Net [8], and Med3D [9]. Therefore, a total of 320 deep networks are trained. To process this workload, a personal compute cluster, Orthrus, is used. The details of Orthrus are provided in the figure below.



The Orthrus cluster is estimated to be able to train the 5 ensembles within about 2 weeks based on sheer FLOPs and the total number of operations per base model architecture in PyTorch. The ensemble is implemented to boost performance of the individual base model by allowing introspection from the meta-model. This builds off the concepts developed in [5] whereby new data folds are iteratively introduced to the ensemble layers so that subsequent layers can learn and correct the previous layer errors. In the aforementioned methods, each subsequent ensemble layer received an additional fold so that the layered meta-models could introspectively learn how the ensemble was learning. In the ensemble architecture designed below, the key differences are that the ensemble layers are flattened and that the folds are introduced in combinations rather than sequentially. This method is preferable for the large dataset because fewer folds are necessary to enable introspective learning.

The proposed ensemble is designed for 5-fold cross validation. Of the 4 training folds, one is reserved for meta-model training. The other three are used to train the base models in all combinations of the folds. Folds that are not included in the training data are used for validation to help prevent overfitting. The base models then predict segmentation masks for the reserved meta-model fold, upon which the meta-model then trains. Lastly, the meta-model inferences the reserved testing fold to generate the standard segmentation metrics: AUROC, Dice, and Jaccard indices. Additionally, the segmented voxels are summed to estimate the Hemothoracic volume. The proposed ensemble architecture generates 7 base models per architecture for each iteration of cross validation, whereas the original methods in [5] would only yield 3 base models per architecture.

The ensemble approach also enables output of a confidence level per pixel, which works toward the maximal goal of the project. Using the reserved meta-model training fold, base model statistics can be collected to weight the respective segmentation capabilities. Then, when inferencing the testing fold, the amount of weighted disagreement between base models can be visualized as a heatmap per pixel per slice. Pixels with little weighted disagreement are considered more reliable than pixels with greater weighted disagreement. This ultimately yields localized confidence levels, which can then be trivially merged via averaging or as a histogram distribution to ascertain global confidence in the segmentation.



5. Testing Plan

- Testing of **preprocessing**: The functions to perform interpolation and 3-D slices will be tested using unit tests. We also plan to eye-inspect the result of the preprocessing against the original file to verify the result.
- Testing of **frameworks, pipelines, and I/O**s: Those API will be tested mainly through unit tests, and their overall function will be verified and debugged during the training.
- Testing of **models**: All models (including the benchmarks and the ensemble) will be validated using Dice and Jaccard indices, as well as the ROC curve.

6. Key Activities & Deliverables

	Activities	**Results/Deliverables**
Minimum	Literature survey for model selection	Draft a list of open-source models with code or architecture description
	Preprocess CT scans (interpolate, make 3d slices)	Interpolate CT scans and convert data to PyTorch tensor type
	Complete pipeline and I/O APIs for the project	Build a network framework consists of Python classes
	Benchmark open-source models	Benchmark existing open-source models measured with Dice/Jaccard
Expected	Design and implement an ensemble algorithm	A program that estimates blood volume (inputs: CT axial scans; output: a value)
	Improve the ensemble algorithm	A program outperforms the benchmark (inputs: CT axial scans; output: a value)
Maximum	Implement a GUI-program for visualization	A program incorporates the framework (inputs: CT axial scans; output: segmentation)
	Incorporate certainty level into our algorithm	A program visualizes confidence (input: CT axial scans; output: heatmaps)

7. Dependencies

Dependency	**Need**	**Status**	**Followup**	**Contingency Plan**	**Planned**	**Hard**

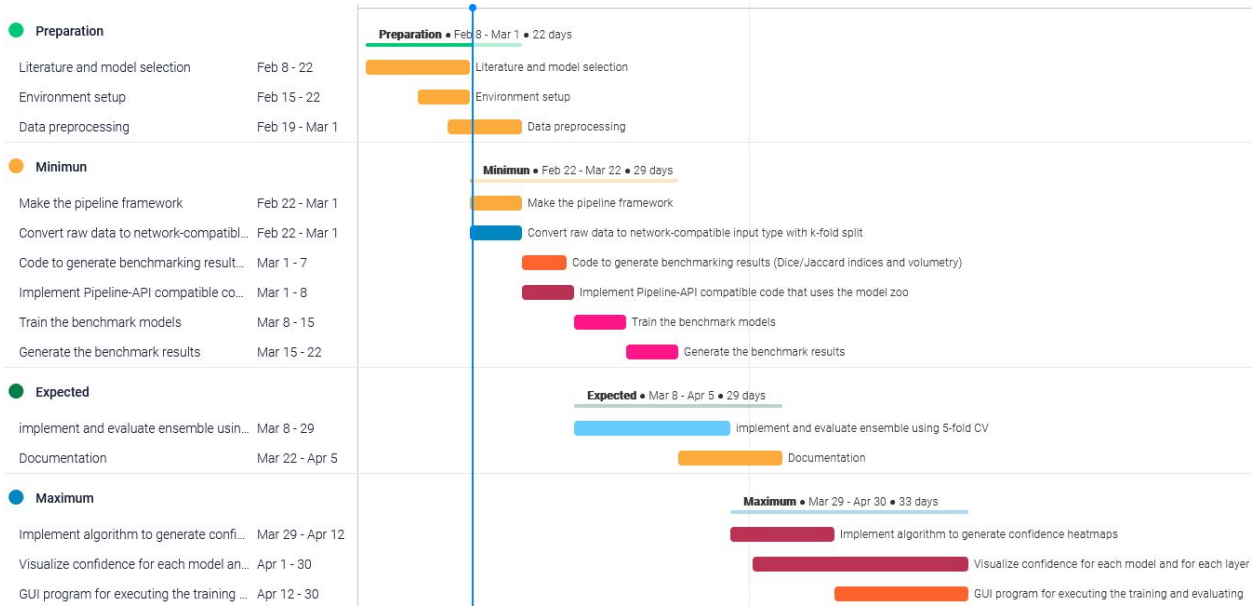
Computing Power	train many models	Orthrus cluster	N/A	Google Cloud Credits	1/28	2/22
CT Scans	train any models	Uploaded to Orthrus	Interpolating to 1mm, building 3 views	N/A	2/18	2/24
CUDA developer.nvidia.com /cuda-zone	GPU interface	Installed	N/A	N/A	1/28	2/22
PyTorch pytorch.org	setup environment	Installed	N/A	Tensorflow tensorflow.org	N/A	N/A
3D Slicer slicer.org	visualize scans	Installed	N/A	Use python packages	N/A	N/A
Open-Source Models	Benchmark and ensemble	Implementing	Run and evaluate	N/A	3/10	3/15

8. Timelines & Project Management

The Timeline of this project consists of two major parallel parts: Project Timeline and Documentation Timeline. The project Timeline includes the designs of the pipeline, I/O, API, GUI, and the building, training, testing, and evaluating our deep learning models. The Documentation Timeline includes the documentation, reports, and other write-ups, which will be done along with our project developments. The details are discussed below:

a) Project Timeline and Management

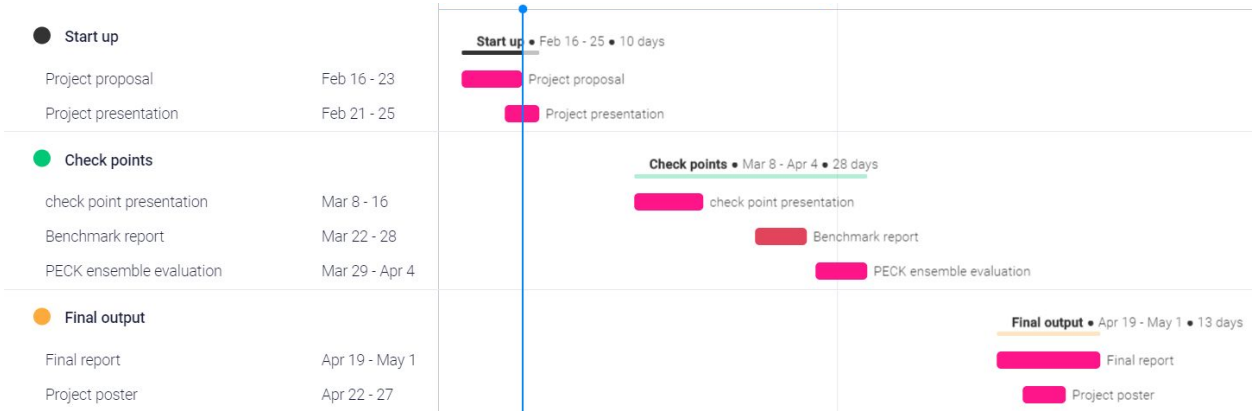
The project involves 4 majors steps. The first step is the design of data conversion, pipeline, and I/O APIs, as well as the result evaluation algorithms, which are to be done before the mid of March. The second step is the four model training to get us the benchmark results, and the third step is our ensemble design and training, which takes the most time of this project, planned to be done by the mid of April. As we have 3 people, those steps are done with parallelism, as shown in the timeline graph. The last step, which is for our maximum deliverables, includes a fancy GUI and training-visualization system to make the training, evaluation, and predicting processes accessible to everyone. To track the progress and manage our project, we use project management software (monday.com) to update tasks, timeline, and progress, as shown below:



Additionally, the main timeline dependency is the ensemble leading into confidence/visualization; if the ensemble runs overtime, then the confidence algorithm and visualization wait until the ensemble is done.

b) Documentation Timeline

The documentation of our API and programs are done along with the implementation and testing processes. Aside from those, our major write-ups are this project proposal, checkpoint report, benchmark report, PEAK ensemble evaluation, poster and final report. The planned timelines for those documentations are shown below:



9. Roles & Responsibilities

a) Team members

- Benjamin Albert
  Undergraduate Junior, BME & CS major, responsible for the design and implementation of PECK ensembles and API, participates in all other works.

- Chang Yan
  Undergraduate Junior, BME & CS & CE major, responsible for the API, GUI design and benchmark evaluation, participates in all other works.

- Gary Yang
  Undergraduate Junior, BME & CS major, responsible for the API design and benchmark training, participates in all other works.

b) Team mentors

- Dr. David Dreizin
  Associate Professor at the University of Maryland School of Medicine, provides the clinical data.

- Dr. Mathias Unberath
  Assistant Professor in the Department of Computer Science at Johns Hopkins University with affiliations to the Laboratory for Computational Sensing and Robotics and the Malone Center for Engineering in Healthcare.

10. Management Plan

a) Meetings

The student team meets with each other everyday through WeChat group or Zoom meetings. Progress is relayed to the mentors every couple of weeks via text and email.

b) Platforms

Codes: We will use Github private repository to collaboratively work on our programs and APIs, and they will be kept private until disclosed by our mentors.

Communications: Communications are mainly done in Wechat Group, Zoom meetings, and emails.

Write-ups: We use private Google Docs to work on reports, presentations and other write-ups collaboratively.

11. References

[1] Zeiler, J., Idell, S., Norwood, S., & Cook, A. (2020). Hemothorax: A Review of the Literature. Clinical pulmonary medicine, 27(1), 1–12. https://doi.org/10.1097/CPM.0000000000000343

[2] Dreizin, D., Zhou, Y., Zhang, Y., Tirada, N., & Yuille, A. L. (2020). Performance of a Deep Learning Algorithm for Automated Segmentation and Quantification of Traumatic Pelvic Hematomas on CT. Journal of digital imaging, 33(1), 243–251. https://doi.org/10.1007/s10278-019-00207-1

[3] Sangster, G. P., González-Beicos, A., Carbo, A. I., Heldmann, M. G., Ibrahim, H., Carrascosa, P., Nazar, M., & D'Agostino, H. B. (2007). Blunt traumatic injuries of the lung parenchyma, pleura, thoracic wall, and intrathoracic airways: multidetector computer tomography imaging findings. Emergency radiology, 14(5), 297–310. https://doi.org/10.1007/s10140-007-0651-8

[4] Dreizin, D., Zhou, Y., Fu, S., Wang, Y., Li, G., Champ, K., Siegel, E., Wang, Z., Chen, T., & Yuille, A. L. (2020). A Multiscale Deep Learning Method for Quantitative Visualization of Traumatic Hemoperitoneum at CT: Assessment of Feasibility and Comparison with Subjective Categorical Estimation. Radiology. Artificial intelligence, 2(6), e190220. https://doi.org/10.1148/ryai.2020190220

[5] Yao, J., Bliton, J., & Summers, R. M. (2013). Automatic segmentation and measurement of pleural effusions on CT. IEEE transactions on bio-medical engineering, 60(7), 1834–1840. https://doi.org/10.1109/TBME.2013.2243446

[6] B. A. Albert, "Deep Learning From Limited Training Data: Novel Segmentation and Ensemble Algorithms Applied to Automatic Melanoma Diagnosis," in IEEE Access, vol. 8, pp. 31254-31269, 2020, https://doi.org/10.1109/ACCESS.2020.2973188

[7] Çiçek Ö., Abdulkadir A., Lienkamp S.S., Brox T., Ronneberger O. (2016) 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: Ourselin S., Joskowicz L., Sabuncu M., Unal G., Wells W. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. MICCAI 2016. Lecture Notes in Computer Science, vol 9901. Springer, Cham. https://doi.org/10.1007/978-3-319-46723-8_49

[8] F. Milletari, N. Navab and S. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," 2016 Fourth International Conference on 3D Vision (3DV), Stanford, CA, USA, 2016, pp. 565-571, https://doi.org/10.1109/3DV.2016.79

[9] Chen, S., Ma, K., & Zheng, Y. (2019). Med3D: Transfer Learning for 3D Medical Image Analysis. ArXiv, abs/1904.00625. https://arxiv.org/abs/1904.00625