Final Project Report for CS 184A/284A, Fall 2023

Project Title: SenNet + HOA - Hacking the Human Vasculature in 3D
Project Number: Canvas Group 35 / Presentation 2023/12/04 Group 3
GitHub Repository: https://github.com/Nitro1231/SenNet-HOA-Hacking-the-Human-Vasculature-in-3D

Student Name(s)

- HyunJun Park, 60978255, hyunjup4@uci.edu
- Rohan Gupta, 31375533, rohang5@uci.edu

1. Introduction and Problem Statement

In the realm of medical imaging and analysis, the precise segmentation of blood vessels in 3D images of human organs has been spotlighted in recent years and has become an area of active research. Unfortunately, the current way of labeling vascular segmentation is a time-consuming and labor-intensive process, as it requires manual labeling. This manual process not only slows down research but also suffers from a lack of generalizability and consistency across different datasets.

Our project, SenNet + HOA - Hacking the Human Vasculature in 3D[1], aims to address this challenge by developing a model that can accurately and efficiently predict vascular structures from advanced Hierarchical Phase-Contrast Tomography (HiP-CT) 3D scan images. To address this, our project utilizes a Convolutional Neural Network (CNN) with an Attention U-Net architecture, known for its efficacy in medical image segmentation tasks. The Attention U-Net model is designed to work by providing precise localizations of the segmented areas. The model's performance is evaluated using the Surface Dice Metric, a metric specifically chosen for its ability to measure the similarity between the predicted and actual segmentation masks. By automating the segmentation process while maintaining high accuracy, our project aims to significantly contribute to the field of medical imaging, aiding in a better understanding of vascular structures and potentially impacting diagnostic and therapeutic procedures.

2. Related Work

The field of medical image segmentation has seen various approaches, mostly using Convolutional Neural Networks (CNNs) due to their effectiveness in handling image data. Historically, approaches like Convolutional Neural Networks (CNNs) have been utilized for image segmentation tasks. One example of this was a Mask Regional Convolutional Neural Network (MaskRCNN) used to segment human kidneys and delineate the kidney into 12 classes, whose results were published in a study in 2021. Results of the study suggested that there is potential for the model to perform, however, it was too challenging to identify specific parts of the kidney because of the low signal-to-noise ratio in the images. Their model consisted of using 3 sequential CNNs, one to get an aligned set of Region of Interest (RoI), one to classify the boundary box from each RoI, and the last to identify the RoI's class and mask. However, for the specific challenge of blood vessel segmentation, the U-Net architecture, a different type of CNN, could be more effective.

---

[1] Yashvardhan Jain, Katy Borner, Claire Walsh, Nancy Ruschman, Peter D. Lee, Griffin M. Weber, Ryan Holbrook, Addison Howard. (2023). SenNet + HOA - Hacking the Human Vasculature in 3D. Kaggle. https://kaggle.com/competitions/blood-vessel-segmentation.

Another significant approach is the U-Net architecture, which is known as effective for medical image segmentation tasks.[2] U-Net's architecture, designed specifically for biomedical image segmentation, offers an advantage in detailed and nuanced tasks like blood vessel segmentation. Our project builds upon these foundational works, focusing on the U-Net model for its superior performance in segmenting complicated structures within 3D medical images. In our project, we decided to develop an improved version of U-Net architecture by adding an attention block to the skip connection, focusing on only important parts and discarding unrelated features of the image.

3. Data Sets

Our project uses a dataset provided by the Kaggle competition. The provided dataset is a collection of high-resolution HiP-CT 3D images of human kidneys and corresponding blood vessel segmentation masks. The dataset is available here (https://www.kaggle.com/competitions/blood-vessel-segmentation/data).

Dataset Composition and Structure:

The dataset is structured into two primary sets: the training and test sets. It contains approximately five kidneys, represented through a series of 2D images. Each image is a top-down TIFF scan, representing a 2D slice of the 3D kidney volume.

- **train/{dataset}/images -** Contains top-down TIFF scans from several kidney datasets, each representing a 2D slice of a 3D volume.
- **train/{dataset}/labels -** Contains blood vessel segmentation masks in TIFF format. The folder contains kidney_1_dense, kidney_1_voi, kidney_2, kidney_3_dense, and kidney_3_sparse subsets, each varying in resolution and segmentation detail.
- **test/{dataset}/images -** Contains the TIFF scans for the test set. These scans may or may not use a different beamline or resolution from the scans used in the training set.

Dataset Size and Accessibility:

- The dataset is quite large, with a total size of 43.52 GB and a total of 14.4K files.
- It is licensed under Attribution 4.0 International (CC BY 4.0).

Image Data Preprocessing and Transformation:

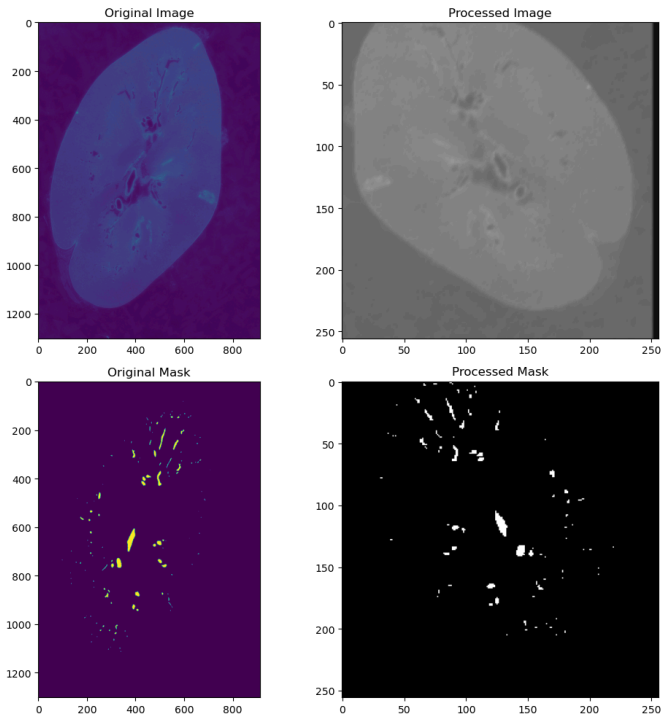
We applied the following transforms to the original images for more accurate train results:

- **Resizing:** Resize the original image to the fixed size of 256x256 pixels sizes. This standardization is essential for ensuring consistent input dimensions for the neural network model.
- **Flipping:** A random horizontal flip and vertical flip are applied to increase the diversity of the training data and help the model learn invariant features.
- **Shift, Scale, Rotate:** Randomly shifts, scales, and rotates the image to increase the diversity of the training data and help the model learn invariant features.
- **Random Cropping:** Ensures that the model focuses on different regions of the image, enhancing its ability to detect features irrespective of their location.

[2] Ronneberger, Olaf, et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." *arXiv.Org*, 18 May 2015, https://arxiv.org/abs/1505.04597.

- **Brightness Variation:** Randomly adjusts the brightness of the image, allowing the model to handle images with varying lighting conditions.
- **Blurring:** Introduces blur to image. This also introduced diversity of the training data.

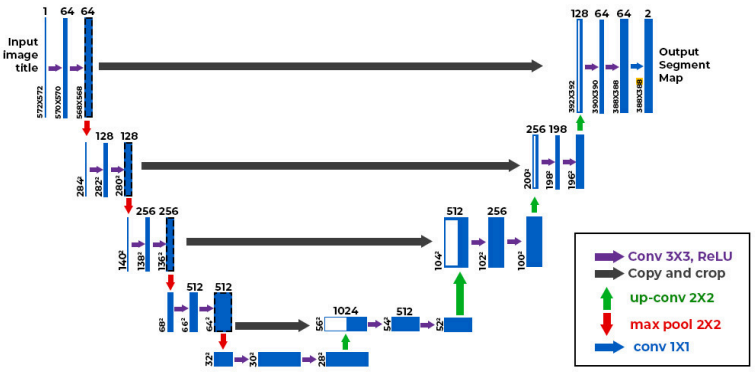
Example of original and processed HiP-CT scan images and blood vessel segmentation masks:



4. Description of Technical Approach

Model: Attention U-Net Architecture

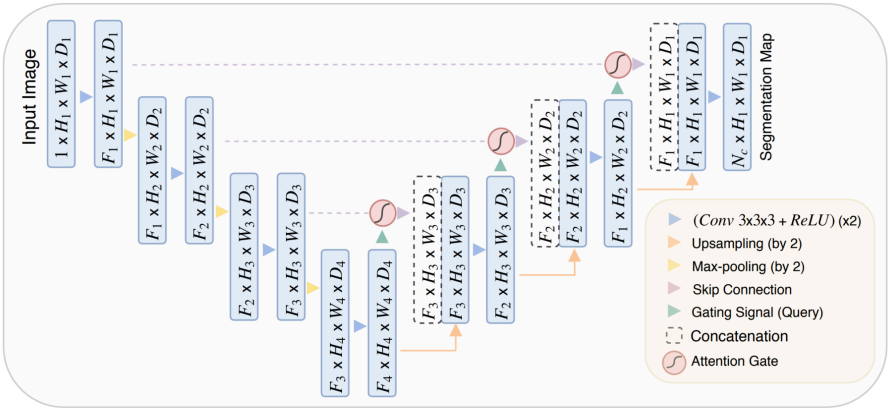
The general structure of our Attention U-Net is to use several encoder layers (downsampling) to gather features from the original dataset, and then use symmetrical decoder layers (upsampling) to determine the segment mask. When upsampling, corresponding encoder outputs will be concatenated before being processed, making segmentation more accurate.



(Image from https://geeksforgeeks.org/u-net-architecture-explained/)

The intuition behind this model is that the decoder layers have more semantic information, such as the area that circles around the object, while the encoder layers likely have more spatial information, such as the specific pixels corresponding to the object.

To improve our model further, we decided to integrate the original U-Net architecture with attention blocks, which allows our model to focus on certain parts of the input data. The attention mechanism is applied to the present and corresponding convolution layer, allowing our model to selectively focus on relevant regions of the input image. This enhances the model's ability to identify and segment target structures with complex backgrounds. Each down convolution layer is performed using 2 convolution blocks, where each block consists of a Conv2d, BatchNorm2d, and ReLU activation function. In between each down convolution layer, we perform one max pool. Each up convolution layer consists of one Upsample, Conv2d, BatchNorm2d, and a ReLU activation function. After one up convolution, we acquire the attention block from the current up convolution and the corresponding down convolution. Each attention block consists of putting the two inputs in a convolution block, concatenating the results, and putting them through another convolution block. This result is then concatenated with the current up convolution and then finally passed to a convolution block, which ends one of our up convolution layers. In our final model, we have 6 down convolution layers and 6 up convolution layers. Consequently, there are 5 attention blocks. Each layer alters the number of channels by a factor of 2, starting with convoluting the initial 3 channel 256 by 256 pixel image to 32 channels.



(Image from "Attention U-Net: Learning Where to Look for the Pancreas," https://arxiv.org/pdf/1804.03999.pdf)

Loss Function: Focal Loss

We picked Focal Loss as our primary loss function instead of using the generic Cross Entropy Loss. This choice was driven by the need to address the class imbalance prevalent in our dataset, where most pixels do not represent blood vessels. It focuses on hard-to-classify examples, giving more weight to incorrectly classified instances. The implementation of Focal Loss in our project is a modified version of binary cross-entropy, providing a balanced approach to handling class imbalance.

Evaluator: Surface Dice Coefficient

The evaluator we used was the surface dice coefficient, which is a variation of the dice coefficient. In the context of segmentation mask similarity, the dice coefficient is modified to take the agreement between the surfaces of two segmented regions instead of their entire volumes. We were given a function implementation of calculating the dice coefficient within the Kaggle Competition, but

the derivation is quite simple. It is 2 times the intersection between the predicted and actual mask divided by their union.

5. Software

We used PyTorch, a machine learning framework, to implement our Attention U-Net. We referenced an Attention U-Net: Learning Where to Look for the Pancreas[3] research and Attention-Gated U-Net notebook[4] posted on the Kaggle site by Aniket Patil for creating scripts necessary for data processing, displaying data, and learning conventional ways to set up our layers and forward methods. We started with defining our custom PyTorch Dataset, which makes it easier to access unorganized original HiP-CT images and segmentation masks. This dataset is also designed to apply image transform on load. We then used Pytorch DataLoader to convert data files into tensors, which we referenced from Homework 3 from this class. We used OpenCV and Numpy to read, resize, and normalize each data image. Following this, we used the package albumentations to augment our images, which consisted of resizing, flipping, rotating, cropping, varying brightness, and blur to create 250x250 grayscale training images. Because neither of us had experience in conventional techniques used for augmenting images, we verified and used the general structure of the augmentation done in the referenced in the Attention-Gated U-Net notebook posted on the Kaggle site. The entire data preprocessing and augmentation was referenced by the model mentioned above. We displayed the images using matplotlib, which we have used in many classes throughout UCI. Our PyTorch attention model had 3 submodels: one for Down Convolution, Up Convolution, and the Attention Gate. We learned of this structure from the reference model mentioned previously. Most convolution blocks consisted of several Conv2ds, BatchNorm2ds, and ReLU. This general structure was adapted from our homework 3 assignments. For the evaluator, we used the Surface Dice Coefficient, which is often used in medical image analysis to quantify the similarity between two surfaces and is particularly useful for assessing the accuracy of segmentation in 3D images. The function to generate the dice coefficient was provided to us in the Kaggle description. We used a variant of Cross Entropy Loss called Focal Loss, which adds a factor to focus training on the hard negatives, which was also referenced from Aniket Patil's posted notebook. For the optimizer, we used the Adam optimizer provided by the PyTorch library with a learning rate of 0.0001. After training, we stored our training/validation results using pandas and our trained model using PyTorch for later use. We displayed our results using matplotlib conventions, and we were provided with a function by the Kaggle competition to generate the run length encoding of our validation kidney results, which was necessary for our submission.

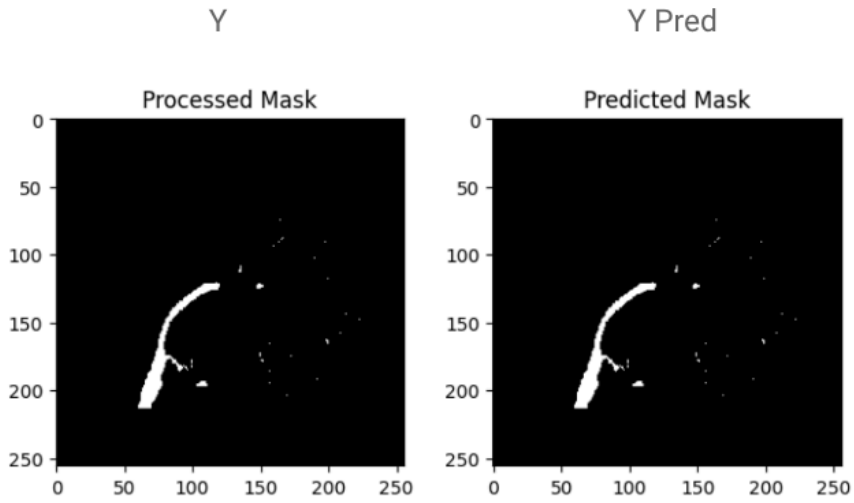
6. Experiments and Evaluation

The foundation of our experimentation lies in a dataset comprising three kidneys (5 folders) for training and two for testing. However, a unique challenge arises with the test kidneys as they lack associated labels or segmentation masks. To overcome this limitation, we chose a cross-validation approach to assess our model's progress. The training dataset was partitioned into 70% for training and 30% for testing. In our initial model iteration, we opted for a U-Net architecture with only two layers to establish a baseline. The model's layers encompassed convolution blocks transforming images from 1

[3] Oktay, Ozan, et al. "Attention U-Net: Learning Where to Look for the Pancreas." *arXiv.Org*, 20 May 2018, https://arxiv.org/abs/1804.03999.

[4] Patil, Aniket. "Sennet+Hoa: Seg.: Pytorch: Attention-Gated U-Net." *Kaggle*, Kaggle, 27 Nov. 2023, https://www.kaggle.com/code/aniketkolte04/sennet-hoa-seg-pytorch-attention-gated-U-Net.

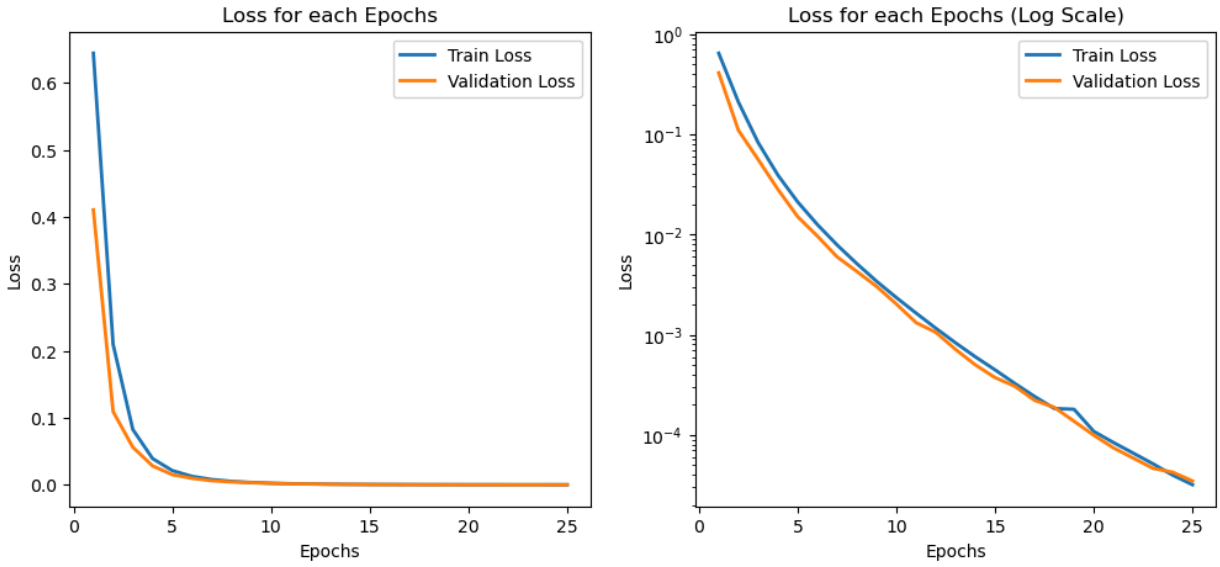
channel to 64 and finally back to 1 channel. As anticipated, this rudimentary model exhibited the poorest accuracy, with a surface dice coefficient hovering around 0.15. With a modest training duration of 15 epochs, both train and validation accuracies indicated the potential for convergence with additional epochs. To further the model's complexity, we expanded the number of layers to five and extended the training duration to 20 epochs. Accompanying this, we chose to reduce channel dimensions by a factor of 2. The image, which initially starts with 1 channel, underwent up-convolution five times to reach 1024 channels and subsequently underwent five down-convolutions with attention to attain 1 channel. Despite this model's significant advancement, achieving a surface dice coefficient of about 0.41, a challenge surfaced during hyperparameter tuning. The prediction mask consistently resembled the processed mask, creating ambiguity in discerning overfitting or underfitting. The similarity between validation and training dice coefficients compounded this because neither showed us signs of what direction to move next. This dilemma prompted a critical evaluation of our five-layer model's capabilities, leading us to believe that we might have reached close to the maximum efficiency for this model, limiting its potential as a more accurate classifier.



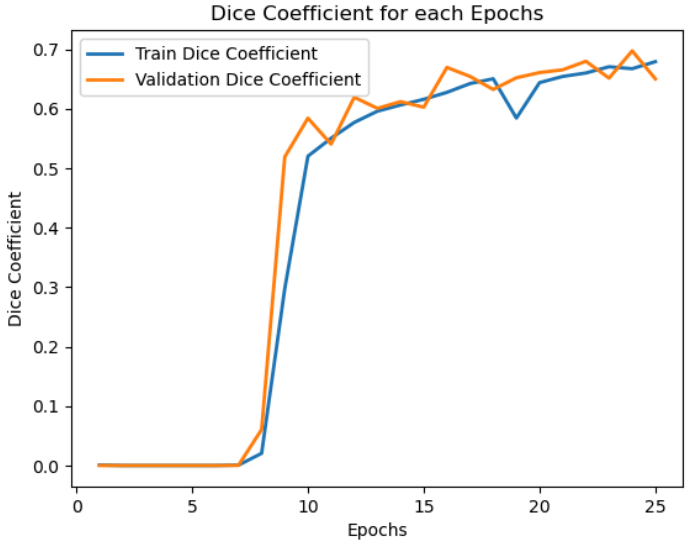
Prediction mask with 5 layer U-Net: 0.003% error rate, but only a 0.41 surface dice coefficient

Despite taking approximately three hours to train, the results indicated room for more complexity. In response, we developed our final model, incorporating six layers, thereby increasing both complexity and training time to over four hours. In crafting this conclusive model, we stuck with a maximum channel size of 1024, but introduced a novel approach in the initial convolution by reducing it to 32 channels instead of 64. A subsequent last up-convolution layer transformed the image from 32 to 1 channel, while the preceding layer adapted to convolute the image from 64 to 32 channels. This strategic alteration aimed to introduce an additional attention layer, hopefully having a strong impact between the initial convolution and the final result of our U-Net. The intention behind this adjustment was rooted in avoiding potential overfitting associated with increasing channels from 1024 to 2048, a consideration that led us to opt for a smaller layer. The model already almost had a 0% error rate, but a coefficient of 0.41 steered us away from increasing the number of channels. The results from this model surpassed expectations, boasting a train and validation dice coefficient of 0.66. Intriguingly, when we displayed the

surface dice coefficient against epochs, it became evident that more convergence was possible even after 20 epochs. Subsequently, we extended the training duration to 25 epochs, requiring five hours, and observed the validation dice coefficients almost reach 0.7. The careful consideration of hyperparameters, layer complexities, and training durations collectively contributed to the evolution of our model, culminating in a robust and fairly accurate segmentation tool for kidney images.



Training and Validation loss with 6 layer U-Net



Prediction mask with 6 layer U-Net: 0.002% error rate, and ~0.7 surface dice coefficient

7. Discussion and Conclusion

Unlike our initial expectations, the model achieved a dice coefficient of around 0.7 and minimal training and validation loss, close to 0. This was a surprising outcome, especially considering the complexity and size of the dataset we worked with. It highlighted the effectiveness of the Attention U-Net in terms of handling complex imaging tasks such as medical image segmentation. We touched on the U-Net architecture slightly in class, but we had never heard of the Attention U-Net architecture before this project. It made the concatenation of the corresponding layers much more complex and was likely a huge factor in our decent results. We were definitely surprised by the results of our model. This was the first time either of us trained a CNN model on such a huge and complicated dataset, and after transitioning from homework 3, we didn't expect the accuracy to be anywhere close to 0%, even if that wasn't our primary evaluator. However, the project also exposed the limitations inherent in current CNN models, particularly in terms of computational resources and time. One severe bottleneck of this problem was the time it took to train the model when it was clear we had to increase the number of layers and the number of epochs. Our final model, which took 5 hours to run, definitely showed indications of having better results if we were to increase the number of layers and epochs. If we had more time to do so, we think this model would reach a dice coefficient close to one. Looking forward, If we were in charge of a research lab, we would focus on exploring ways to optimize the efficiency of our model. Investigating methods to reduce training time without compromising accuracy would be a key area of interest. Additionally, we would dive into the feature retrieval in each individual layer to understand how each layer affects our final mask. To do this, we would need to learn how to understand and translate the channel's weights into a representation of the features it can find. Doing so would allow us to tune individual layers and likely find evidence toward increasing or decreasing the number of layers or convolutions. These approaches, while ambitious, could significantly advance the field of medical imaging and beyond, leveraging the full potential of CNNs in complex segmentation tasks.

8. A separate page on *Individual Contributions*

HyunJun Park

In our project, I believe that Rohan and I achieved a well-balanced distribution of tasks and responsibilities. As a team, we spent considerable time discussing the problem and brainstorming potential solutions. Regarding individual contributions, I felt that both of us did best for our responsibilities, and equality contributing to the project. Personally, I believe that I focused more on the dataset and data pre-processing, while Rohan mainly focused on model improvement and preparing the final project presentation. This division allowed us to leverage our individual strengths while maintaining a collaborative effort throughout the project. We worked together on improving the code, model parameter tuning, and finalizing both the written and oral presentations, ensuring a balanced contribution from each of us.

Rohan Gupta

I believe HyunJun and I contributed an equal amount of work towards this project. We spent time together to understand this problem and look for potential solutions on both the Kaggle site and through outside resources. Upon deciding on our model, HyunJun was primarily responsible for processing and augmenting the dataset and formulating the structure of the notebook and the libraries we used. Most of the parts I worked on consisted of designing and tuning the Model and the corresponding training, loss, and evaluator functions. I was primarily looking at our evaluators and graphs and advising/implementing what step we should take next. We both spent an equal amount of time writing the project proposal, presentation slides, and the final report.

References

Adejare, Adebayo. “EDA + TF-Keras Starter - Vasculature Segmention.” *Kaggle*, Kaggle, 27 Nov. 2023, https://www.kaggle.com/code/adebayo/eda-tf-keras-starter-vasculature-segmention.

Oktay, Ozan, et al. “Attention U-Net: Learning Where to Look for the Pancreas.” *arXiv.Org*, 20 May 2018, https://arxiv.org/abs/1804.03999.

Overgaard Lauersen, Mathilde, et al. “Kidney segmentation for quantitative analysis applying MASKRCNN architecture.” *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2021, https://doi.org/10.1109/ssci50451.2021.9660052.

Patil, Aniket. “Sennet+Hoa: Seg.: Pytorch: Attention-Gated U-Net.” *Kaggle*, Kaggle, 27 Nov. 2023, https://www.kaggle.com/code/aniketkolte04/sennet-hoa-seg-pytorch-attention-gated-U-Net.

Ronneberger, Olaf, et al. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” *arXiv.Org*, 18 May 2015, https://arxiv.org/abs/1505.04597.

rupert ai. “The U-Net (Actually) Explained in 10 Minutes.” *YouTube*, YouTube, 5 May 2023, https://www.youtube.com/watch?v=NhdzGfB1q74.

Yashvardhan Jain, Katy Borner, Claire Walsh, Nancy Ruschman, Peter D. Lee, Griffin M. Weber, Ryan Holbrook, Addison Howard. (2023). SenNet + HOA - Hacking the Human Vasculature in 3D. Kaggle. https://kaggle.com/competitions/blood-vessel-segmentation.