Project Report by Group Acorn: Machine Learning in Medical Image Processing

Automated Coronary Artery Segmentation and Coronary Tree Graph Representation

Ons Bousbih, Anton Segeler

1. Introduction

Coronary artery disease (CAD) is a common type of heart disease and one of the leading causes of mortality worldwide [1]. This condition is caused by the narrowing of the coronary arteries (called stenosis) due to the accumulation of fat, cholesterol, and other substances. These deposits form atherosclerotic plaques in the vessel walls, can obstruct the blood flow, and may lead to myocardial ischemia or infarction if the artery becomes completely occluded [2].

Hence, accurately segmenting the coronary arteries from medical images is a crucial step in the diagnosis of CAD. It allows clinicians to evaluate the vessel structure and detect stenosis. However, manual segmentation is a time-consuming task that is subject to inter-clinician variability [3].

In this context, machine learning approaches can assist clinicians in overcoming these challenges by providing a segmentation tool, that requires minimal human intervention or by enabling a fully automated and reproducible coronary artery segmentation pipeline. Automating segmentation allows physicians to dedicate more time to analyzing complex cases rather than spending hours on manual annotations. Additionally, it reduces human errors and variability, enhancing the reproducibility of results.

In the scope of this project, we focused on building an automated pipeline for coronary artery segmentation from computed tomography angiography images (CTA). We worked specifically on developing a solution for two main tasks: firstly, segmenting coronary arteries from CTA and secondly, constructing a tree graph representation from the previously produced segmentation masks. This graph-based approach serves as a foundation for further analysis of the coronary artery tree, enabling the extraction of key metrics such as branch depth and length, as well as the automatic detection of anomalies in cases of incomplete or irregular graph structures.

ons.bousbih@campus.tu-berlin.de (O. Bousbih);
segeler@campus.tu-berlin.de (A. Segeler)

2. Automated Coronary Artery Segmentation

In this section, we focus on finding a suitable approach for the segmentation of the coronary artery from CTA scans. We explore the different steps and experiments followed to build an adequate segmentation pipeline and present different results obtained in the validation process.

2.1. Methods

2.1.1. Model Requirements

When selecting a model architecture for this task, several key factors were considered. Rather than developing a model from scratch, we opted to leverage existing research and focus on optimizing the model for our specific dataset and task.

First, the model needed to be specialized for medical imaging. Medical data, such as computed tomography angiography (CTA) scans, exhibit unique structural characteristics that require tailored models [4].

Second, the model had to be designed for 3D data processing. While some models can handle 3D data by processing individual 2D slices, this approach results in a loss of spatial context, making it suboptimal for segmenting complex vascular structures [4].

Third, we considered the impact of Transformer-based architectures, as they have significantly improved image processing in recent years. Given that our task involves analyzing branching structures, a Transformer-based approach was particularly appealing due to its ability to capture long-range dependencies and hierarchical patterns in the data [5].

Lastly, the model needed to be computationally efficient, capable of running on smaller GPUs without significant performance degradation.

2.1.2. Architecture

A model that meets all these criteria is UNTER, short for UNEt TRansformer, which is specifically designed for medical image segmentation [5].

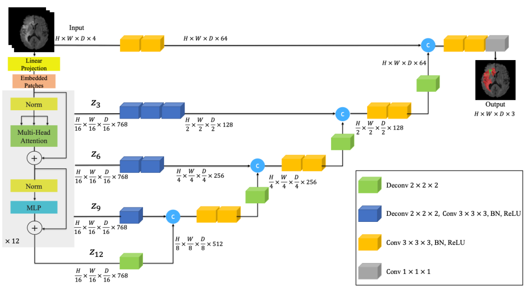
Input
$H \times W \times D \times 4$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
$\times 12$
z_3
z_6
z_9
z_{12}
Output
$H \times W \times D \times 3$
Deconv $2 \times 2 \times 2$
Deconv $2 \times 2 \times 2$, Conv $3 \times 3 \times 3$, BN, ReLU
Conv $3 \times 3 \times 3$, BN, ReLU
Conv $1 \times 1 \times 1$

Figure 1: UNETR architecture [5]

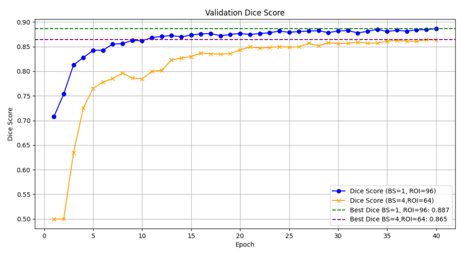

Figure 2: Comparison of validation Dice scores for different model configurations: reducing batch size to 1 versus reducing the Region of Interest (ROI) to 64 × 64 × 64.

As seen in Figure 1, taken from the original UNETR paper, the network has a U-shaped architecture, consisting of two main components: the Transformer encoder and the CNN-based decoder.

Unlike classical UNET [6], the transformer-based encoder allows the network to capture long-range dependencies more effectively.

The encoder divides the input volume of size 96 x 96 x 96 into non-overlapping 16 x 16 x 16 patches and embeds them into a sequence of tokens. The tokens pass afterwards through multiple self-attention layers, which enables the model to learn global contextual features.

The decoder has a similar structure to a UNET decoder. It reconstructs the final segmentation mask by progressively upsampling the feature maps.

Another UNET-inspired element is the skip connections that link the encoder and decoder in several stages. These connections help preserve the spatial details that are lost in deeper layers.

Initially, we aimed to stay as close as possible to the architecture presented in the UNETR paper by Hatamizadeh et al. [5] to leverage the findings already established in their work. In the Experiments section, we further investigate how specific architectural choices and hyperparameter selection impact performance on our dataset.

To run the model within our GPU capacity of 11 GB, we had to make certain compromises, as the exact parameter choices from the paper exceeded our computational resources. Therefore, we investigated which parameter adjustments would allow the model to run within our computational limits while minimizing performance degradation.

For the first approach, we reduced the batch size from the original value of 4, as used in the paper, to 1. While this increased the training time, we expected it to have a minimal impact on model performance. Additionally, in the Experiments section, we analyze how increasing the learning rate while keeping the batch size low influences model performance. As suggested by Smith et al. [7], this strategy should yield comparable performance to a low learning rate with a larger batch size.

For the second approach, we maintain a batch size of 4 but reduce the Region of Interest (ROI), which refers to the cropped section of the input image. This step was necessary because the UNETR architecture requires a fixed input size. The original paper uses an ROI of 96 × 96 × 96, which we decreased to 64 × 64 × 64.

As shown in Figure 2, the validation Dice score is significantly better with the first approach. Therefore, we selected this configuration for future implementations.

We initially fixed a selection of hyperparameters, which will be discussed in detail across the different experiments.

The model was trained using the AdamW optimizer with weight decay regularization, using the following parameters: learning rate (Lr) = 10^{-4} and weight decay = 10^{-5}. Each training session consisted of 40 epochs on the training dataset, as described later in the pre-processing section.

Validation was performed using a sliding window inference technique, which processes the input by extracting overlapping 3D patches of size (96, 96, 96). These patches are passed through the model, and the final prediction is obtained by merging the outputs using an average-weighted approach in the overlapping regions.

2.1.3. Data Analysis and Pre-Processing

Raw CTA data can generally be inhomogeneous, noisy, and of variable intensity distributions. Before training a neural network for coronary segmentation we must analyze the data and apply pre-processing steps to standardize the available dataset and match it to the input requirements of the neural network.

For this project, we worked with a dataset of 800 CTA 3D images, of dimensions in the range of 512 x 512 x 270 voxels. We initially split the dataset as follows: 70% for training, 25% for validation, and 5% for testing. In the

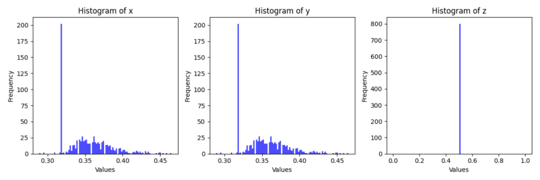

Figure 3: Histogram of voxel spacing of the dataset

final phase of this project, we had access to an additional 200 CTA images without annotations for the final testing.

As shown in Figure 3, the voxel spacing in our dataset is not uniform: the x and y spacing varies from 0.28 mm to 0.47 mm, and the z-spacing is constant equal to 0.5 mm. In order to standerdize the data, we chose to initially downsample the images to a voxel size of 1x1x1 mm^3. This resampling procedure ensures consistency across the dataset, which makes it easier for the neural network to learn meaningful features without being affected by the varying resolutions.

We further analyzed the data by plotting the histogram of voxel intensities shown in Figure 4. It reveals three main peaks corresponding to air (around -1000 HU), soft tissues and blood (around 0 HU), and the third (around 300 HU) corresponds to the contrast-enhanced vessels, as the coronary artery attenuation in CTA is usually in the range of 300-400 HU [8].

To improve segmentation, intensity values were clipped to the range of [-750, 500] HU. We remove the air by choosing a lower bound of -750 HU and preserved the tissue of interest and removed noise by fixing the upper bound at 500 HU.

We further normalized the intensities by scaling them to the range [0,1] for consistency across scans. Foreground cropping was performed to remove unnecessary background regions and focus only on relevant structures. Patches were also sampled during training to extract four patches of size 96×96×96 per scan, finally padding of crops to the size of 96×96×96 is applied to ensure that all crops are of uniform size. These steps ensure compatibility with the initial input size of the UNETR architecture.

Additionally, some probabilistic data augmentation transformations were applied to the training set to enhance the model's robustness: random flipping along each axis (10% probability), 90° rotations (10% probability), and intensity shifts by ±0.1 (50% probability).

2.1.4. Post-processing

The post-processing consists of two main steps. First, since the output predictions are downsampled to a resolution of 1x1x1 mm^3 compared to the original images, they need to be upsampled back to the initial resolution.

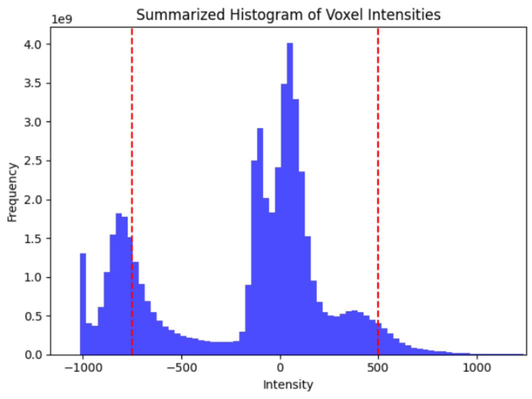

Figure 4: Histogram of intensities in the dataset

Second, as the raw predictions include discontinuities in the artery structure (see Figure 15), disconnected components have to be either removed or connected to the main structure. In our present work, we focus rather on the first method, which is removing the disconnections. Two different methods were explored here: keeping only large components and removing structures smaller than a certain threshold.

As shown in Figure 5, we computed the mean Dice score on the validation dataset using the Connected components method while varying the number of components kept k. The best performance was obtained with k=4.

A similar analysis was performed using the second method, shown in Figure 6. Objects smaller than the threshold size s (in voxels) were removed. The best mean dice score is obtained with the value s=64.

Considering the optimal values for k and s, both methods produced similar Dice scores of 0.867 and 0.866, respectively. The second method was adopted for the final submission as it provides a better global performance across other evaluation metrics.

2.2. Experiments

2.2.1. Voxel Spacing

Since we are working with a fixed-size input, the voxel spacing effectively determines the size of the crop taken from the original CT scan to produce our segmentation. This led us to investigate whether increasing the voxel spacing compared to the original spacing of 1.0 × 1.0 × 1.0 would improve the segmentation performance. The hypothesis was that a larger voxel spacing would allow the model to capture more contextual information from the CT scan, although at a slightly reduced resolution.

We selected a voxel spacing of 1.5 × 1.5 × 2.0, and the

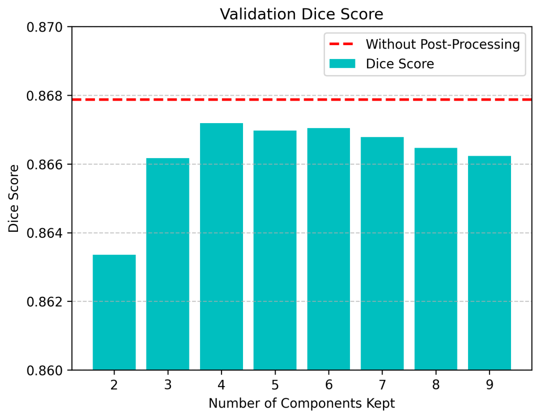

Figure 5: Validation dice scores of the number of kept components k

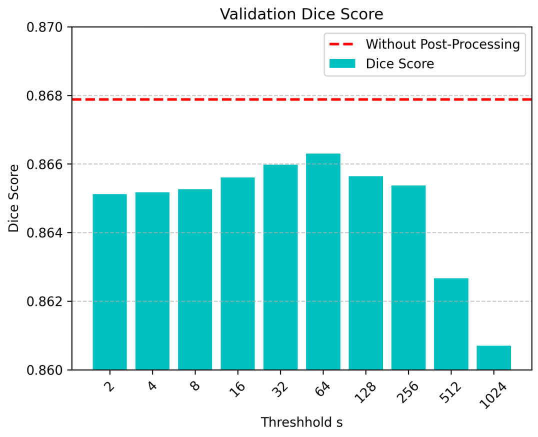

Figure 6: Validation dice scores of the threshold size s

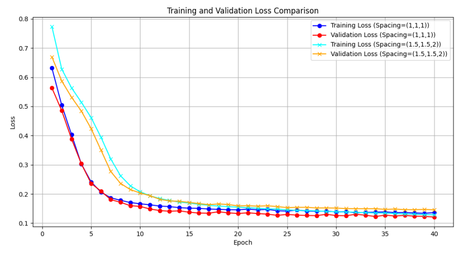

Figure 7: Training and validation losses with different voxel spacing

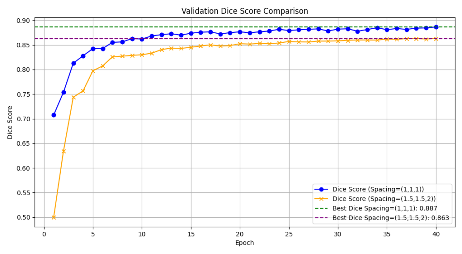

Figure 8: Validation Dice score with different voxel spacing

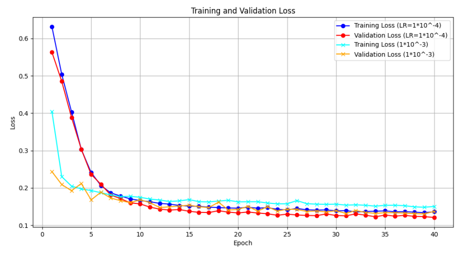

Figure 9: Training and validation losses different LR values

results are shown in Figure 7 and Figure 8. The figure indicates that the original voxel spacing performs better in both training and validation. This is likely due to our cropping procedure, which increases the probability that the foreground is in the center of the crop. As a result, parts of the coronary artery are rarely cropped out.

It is important to note that our validation procedure ensures the entire CT scan is segmented, as we employ a sliding window approach.

2.2.2. Learning Rate

As mentioned earlier, it is relevant to analyze our choice of learning rate, as we decreased the batch size to accommodate our limited computational resources. In our experiment, we tested the original learning rate from the paper 10^{-4} and an increased learning rate of 10^{-3} to compensate for the reduced batch size.

As shown in Figure 9 and 10, the higher learning rate performs better during the early epochs of training but appears to be unstable later, ultimately resulting in a lower best validation Dice score compared to the lower learning rate. Based on these findings, we decided to use a learning rate of 10^{-4}.

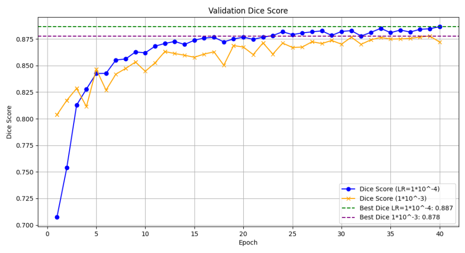

Figure 10: Validation Dice score different LR values

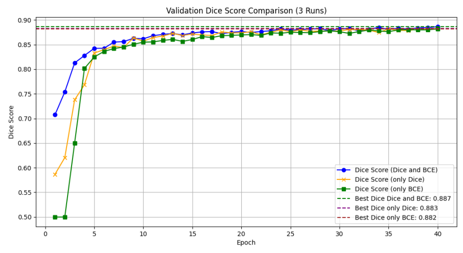

Figure 11: Validation Dice score with different loss functions

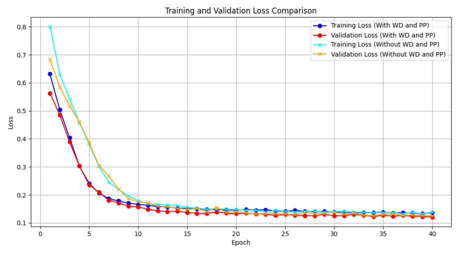

Figure 12: Training and validation losses with and without pre-processing and regularization

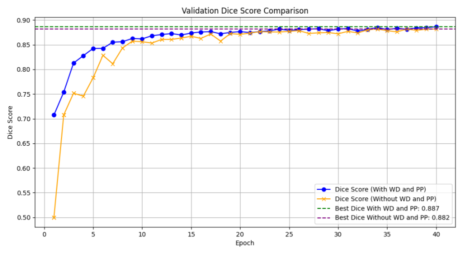

Figure 13: Validation Dice score with and without pre-processing and regularization

2.2.3. Loss Function

For the choice of loss function, we initially planned to use the Dice loss, as literature suggests it is well-suited for handling imbalanced datasets with small foreground objects like ours [9]. In the UNETR paper, an equally weighted combination of Dice loss and cross-entropy loss is used. Therefore, we tested whether Dice loss, binary cross-entropy, or an equally weighted combination of both would yield the best performance.

The validation Dice score for the three loss functions is plotted in Figure 11. While the weighted combination of both losses achieves a slightly higher best validation Dice score, the differences in performance are minimal. Consequently, we decided not to further investigate other weighted combinations of these losses and adopted the equally weighted approach in our model.

In the Future Work section , we discuss additional loss functions that might be applicable to our task.

2.2.4. Detection of Overfitting

When analyzing the evolution of training loss, validation loss, and their relationship over training epochs, we did not detect any signs of overfitting. We hypothesized that this could be due to the original model incorporating weight decay and several data augmentations in the pre-processing stage, including random cropping, random flipping, random rotations, and random intensity shifts, which help improve generalization. To test this, we examined the loss evolution after removing these two features.

Figure 12 and 13 present these plots, showing that without weight decay and data augmentation, both training and validation loss decrease more slowly. However, no overfitting is observed, and both approaches appear to plateau at the same performance level.

In conclusion, we suspect that the large dataset contributes to the strong generalization performance of the model. Further investigation, such as training on a smaller dataset, exploring alternative learning rate schedulers, or varying the model architecture, is needed to better understand why our model does not exhibit overfitting.

2.3. Results

2.3.1. Quantitative Results

For a quantitative analysis of our approach, we computed the following metrics on our test dataset: mean Dice score (F1-score), mean Hausdorff distance, mean Jaccard score, and mean centerline Dice [10].

We considered four different model configurations,

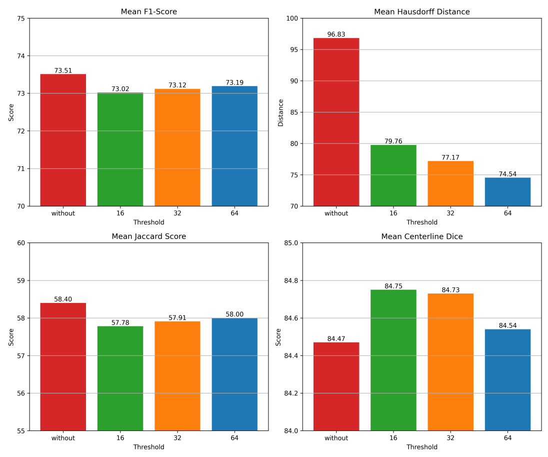

Figure 14: Metrics results for the Coronary artery segmentation task

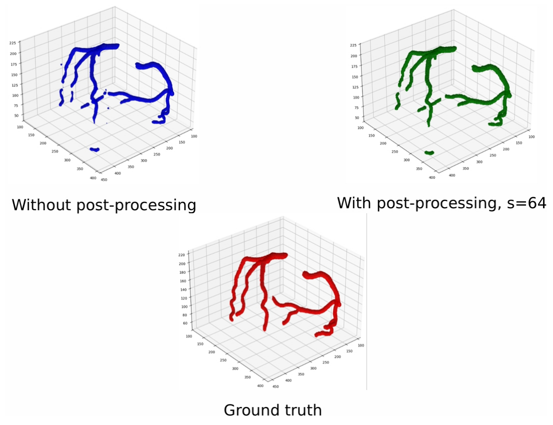

Figure 15: Examples of segmentation masks

each applying a post-processing step that removes objects smaller than a threshold s, where $s \in [0, 16, 32, 64]$. The results of this experiment are shown in Figure 14. The figure indicates that no single configuration performs best across all four metrics.

The configuration without removing small objects ($s = 0$) achieves the highest Dice score and Jaccard score. This is likely because some of the small objects removed by other configurations correspond to segmented regions in the ground truth that are disconnected from larger components.

The configuration with $s = 64$ outperforms the others on the Hausdorff distance metric. This is plausible, as this configuration removes the smallest disconnected errors that appear in the model's predictions.

For the mean centerline Dice, all configurations perform within a range of 0.28, with $s = 16$ achieving the best performance. This is expected, as this metric evaluates how well the predicted centerline aligns with the ground truth [10]. Since the post-processing configurations primarily affect small structures, they do not significantly alter the centerline.

Ultimately, the choice of configuration depends on the user's preference, depending on the relevance of each metric to their specific segmentation task.

Please note that the Dice scores in this section differ from those presented in the Methods section. This is because the Dice scores here refer only to the foreground dice, whereas in earlier sections the reported values represent the mean Dice score for both foreground and background. However, optimizing either of these two losses effectively addresses the same problem.

2.3.2. Qualitative Results

A qualitative example of our model's results is shown in Figure 15, along with the prediction before post-processing and the ground truth.

Overall, the model accurately segments most parts of the coronary artery tree. However, several issues can be observed. First, some artery endpoints are disconnected from the main structure. In other examples, this issue often occurs when the artery becomes thin and twisted. During post-processing, some of these disconnected parts are removed due to their small size, while others remain in the final prediction.

Furthermore, our post-processing step effectively removes small prediction errors that randomly occur in the segmentation. However, if these errors are too large, they may persist in the final mask. As a result, selecting the threshold parameter s for our model involves the trade-off already mentioned in the quantitative results section : either relevant parts of the segmentation are removed, or prediction errors remain in the final output.

3. Coronary Tree Graph Representation

Starting from the masks predicted by our segmentation model, this task focuses on generating a graph representation of the coronary tree. This representation provides additional information about the size, length, and depth of the coronary tree, aiding medical technicians in detecting anomalies.

In this section, we provide an overview of the pipeline used to generate the graph and present insights from experiments that led to an improved graph representation.

3.1. Pipeline (Methods)

Step 1: Skeletonization

In the first step, the mask is reduced to a one-pixel-wide representation. This is achieved by iteratively removing pixels at the border of the mask, ensuring that connectivity is preserved. We used the implementation available in the Scikit-image library for the skeletonization algorithm described by Lee et al. [11].

Step 2: Keep Two Largest Components

To separate the coronary tree into its left and right components, we apply the same method as in post-processing to retain only the two largest connected components. Further methods for reconnecting disconnected parts of the coronary tree caused by segmentation errors are beyond the scope of this project but will be discussed in Future Work.

Step 3: Graph Construction

From the skeletonized components, we construct a graph by assigning a node to each voxel in the skeleton and creating an edge between two nodes if they are adjacent. For all edges, we also calculate the length as the Euclidean distance in world space.

Step 4: Split the Graph

In this step, we separate the graph into its two connected components. Steps 4 through 8 are performed individually on each one of the graphs.

Step 5: Simplify

Since the final graph representation should contain nodes only at endpoints and bifurcation points, we simplify the graph by removing all nodes with a degree of 2. This is done iteratively by removing such nodes and connecting their neighboring nodes with an edge. In the section Experiments, we propose and test different methods for calculating the length of the newly created edges.

Step 6: Merge Nodes

Upon analyzing our graphs at this stage, we observe that they contain too many nodes (and edges) compared to the ground truth. An example of this can be found in the appendix. This often occurs due to small segmentation errors (bumps) that, after skeletonization, result in additional branches.

To address this issue, we merge nodes if they are connected by an edge with a length smaller than a threshold d. The new node's position is computed as the mean position of the merged nodes, and the length of the deleted edge is distributed equally among all edges connected to the new node (these are all edges that were originally connected to either of the deleted nodes).

Step 7: Simplify

Since merging nodes (Step 6) can introduce additional nodes with a degree of 2, we apply the same simplification process as in Step 5.

Step 8: Determine Root Nodes

The root node represents the origin of the coronary artery tree and serves as the starting point of blood flow [12]. Since the ground truth of our graph representation includes information on the root node for each graph —necessary for computing the depth of the coronary artery tree- we also require a method to determine the root node for both the left and right coronary artery trees.

In the Experiments section, we evaluate different methods to determine which approach minimizes the difference between our computed depth and the ground truth.

Step 9: Direct the Graph

Once the root node is determined, we direct the graph to follow the direction of blood flow. This is implemented using a depth-first search [13] starting from the root node.

Step 10: Merge the Graph

In the final step, the left and right coronary artery tree graphs are merged to form the final Coronary Tree Graph Representation. The resulting graph is then converted into a JSON format to match the desired structure of the ground truth.

3.2. Experiments

3.2.1. Edge Length

For calculating the length of an edge, we propose two different methods. In the first method, we compute the Euclidean distance between the nodes of an edge after all simplification steps. In the second method, we use the initial edge length calculated in Step 3. Whenever a node of degree 2 is removed, we sum the lengths of the deleted edges and add it to the newly created edge. This approach preserves the original skeleton length of the coronary artery.

As shown in Figure 16, the skeleton method results in a lower mean absolute error in graph size for both the left and right coronary trees. This indicates that the skeleton method better preserves the coronary artery structure compared to calculating a straight-line Euclidean distance. However, when evaluating intra-class correlation, the results are less conclusive. While the skeleton method performs better on the right coronary tree, the Euclidean distance method yields a higher correlation on the left tree.

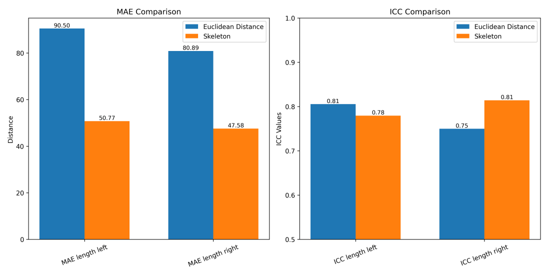

Figure 16: Metrics comparison for different edge length computation methods

3.2.2. Merge Nodes

We conducted an experiment to determine the optimal threshold value d for merging nodes that are close to each other. For $d \in 3, 5, 7, 9$, we calculated the evaluation metrics considered for this project, as shown in Figure 17. We concluded that no single value of d optimizes all metrics. Both $d = 5$ and $d = 7$ yielded the best results for certain metrics, but we decided to choose $d = 5$, as we aimed to avoid oversimplifying the graph unnecessarily.

It is important to note that these experiments are based on optimizing the metrics for graph depth and graph size. This optimization comes with the trade-off of potentially removing edges in locations where they should be preserved. Ultimately, it is up to the user to decide whether to prioritize metric optimization or allow the graph to retain more details in terms of size and depth, preserving potentially relevant structural information.

3.2.3. Determine Root Node

For determining the root node, we conducted experiments on three different methods. In the first method, we simply select the leaf node with the highest z-coordinate in each tree. This approach is based on the observation that root nodes are often located in the upper parts of the CT scan.

The second method, which was adapted from another group in this project, calculates the distance of all leaf nodes to the center point of the highest slice in the CT scan and selects the closest one. This method builds on the first approach by incorporating the additional assumption that root nodes are often positioned closer to the center of the scan.

For the third method, we followed an algorithmic approach. We observed that root nodes are rarely located at the end of the longest path in the tree, as these paths often connect one leaf node to another. Using a heuristic approach, we found that the root node is often near the center of this longest path. Therefore, we first identify the longest path in the tree, determine its center node

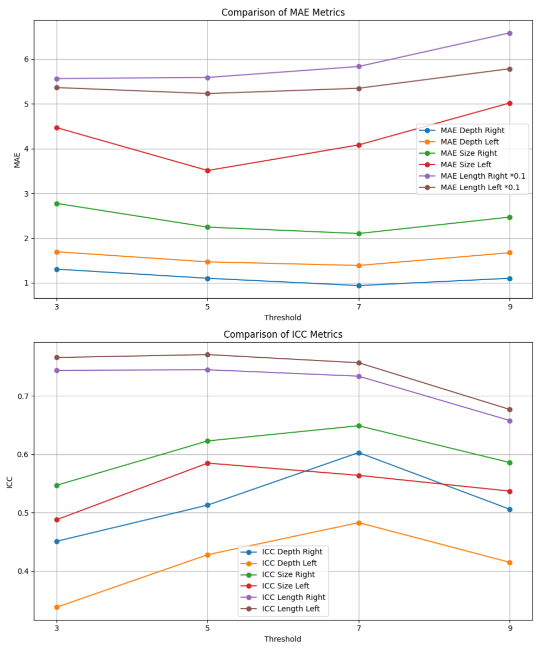

Figure 17: Evaluation metrics comparison for merging nodes with a varying threshold d

(the node closest to the middle of the path), and finally select the closest leaf to this center node as the root node.

In the appendix, a figure presents the evaluation metrics for all three methods. The results show that the Highest Leaf method performs best for all but one metric. Based on this, we selected this method for our pipeline. In the Future Work section, we will discuss further approaches for determining the root node either algorithmically or based on its position in the CT scan.

3.3. Results

3.3.1. Quantitative Results

Due to the merging of close nodes, we were able to achieve graphs that are structurally similar to the ground truth across multiple metrics. Figure 18 presents the metrics achieved by our approach.

As mentioned in the Experiments section, optimizing based on these metrics is a reasonable approach. However, retaining a graph with more nodes and qualitatively analyzing issues based on specific examples is also a valid strategy to achieve a graph representation that better aligns with the structure of the ground truth, at the cost of compromising the optimization of certain metrics. Due to time constraints, we chose our implemented approach, as it requires less manual intervention.

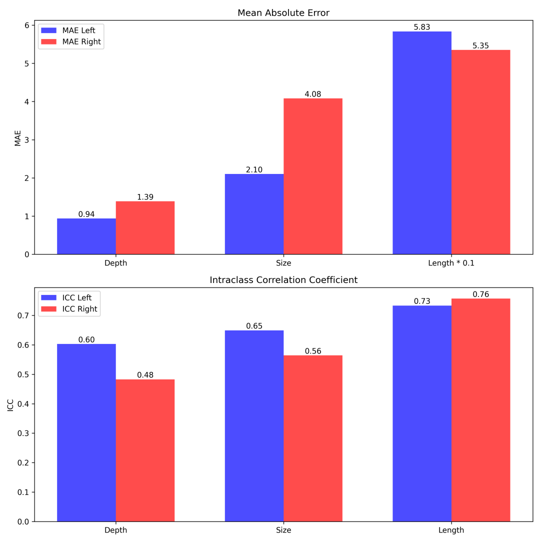

Figure 18: Metrics results for the tree graph representation task

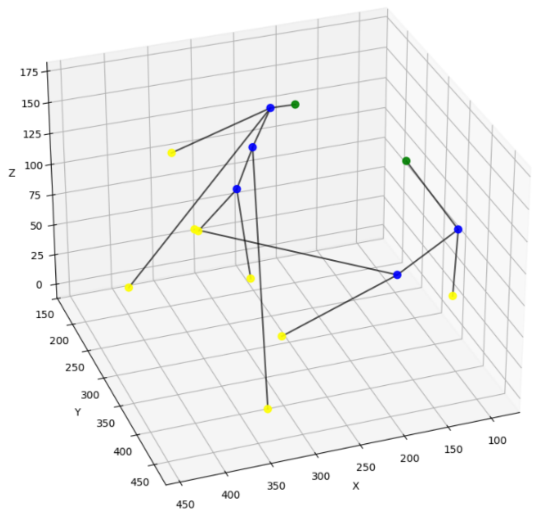

Figure 19: Example of graph prediction

3.3.2. Qualitative Results

Looking at an example of our predicted graph representation in Figure 19, the structure appears very similar to the ground truth in Figure 20. Since our approach is optimized based on specific metrics—graph size, graph depth, and graph length—these values closely match those in the ground truth. However, as a consequence, the graph is over-simplified in some areas while containing unnecessary nodes in others.

Furthermore, due to errors in our segmentation model, parts of the coronary artery that were disconnected in the prediction - and subsequently removed when retaining only the two largest components - are missing from the final graph representation.

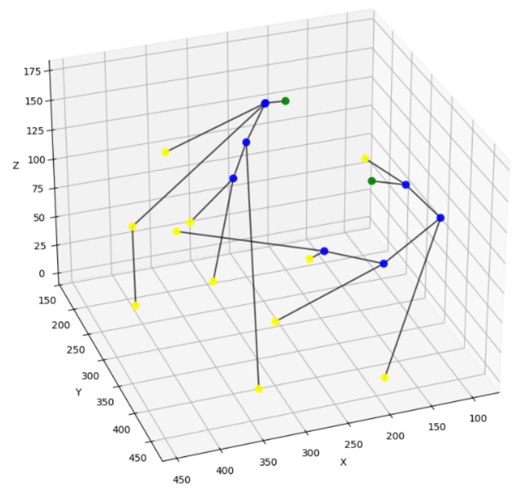

Figure 20: Example of a ground truth graph

4. Discussion

4.1. Challenges

While completing this project, we encountered several challenges. One of the main challenges in Automated Coronary Artery Segmentation was the limitation in computational power. Segmentation models that incorporate transformer-based architectures tend to have a high number of parameters, which required the modifications discussed in Methods. Additionally, this limitation restricted the number of experiments we were able to conduct.

Another challenge, was the fact that we worked with an already high-performing transformer-based model like UNETR. While focusing on the Automated Coronary Artery Segmentation, we had the opportunity to gain a deeper understanding of the UNETR architecture and its optimization pipeline. However, we struggled to reach a big improvement compared with our initial training results. Testing simpler architectures such as UNET, alongside UNETR, could have been an interesting approach to reach more significant improvement and compare the learning scheme of both architectures. Unfortunately, due to time and resources limits this was not possible for our group.

Additionally, while this project provided us with some interesting knowledge about the heart anatomy, coronary artery structure, and coronary artery disease, our initial lack of medical knowledge and experience with medically-related machine learning techniques slowed our progress at the start in problem understanding and medical data analysis phases.

4.2. Future Work

Future work for this project could focus on improving post-processing for Automated Coronary Artery Segmentation to eliminate the trade-off mentioned in the results. One possible approach would be to remove objects based on additional conditions, such as the minimum distance to larger connected components above a certain size.

Additionally, the model could be optimized using alternative loss functions, such as clDice [10], to improve metrics beyond the Dice score. We conducted preliminary experiments with clDice, but due to its high computational cost, further exploration of this approach was beyond the scope of this project.

For the Coronary Tree Graph Representation, future work could focus on reconnecting components that are part of the coronary artery but were disconnected during segmentation. Other groups have proposed promising approaches, where the minimum distance to the closest object above a certain size is calculated, and an edge is created along this minimal-distance path.

Finally, there is significant potential for improving root node selection. One possible approach is statistical modeling, where the mean position of all root nodes for the left and right trees is computed from the training data, and the predicted node closest to this position is selected. Another approach is algorithmic, in which the structural patterns of the coronary tree graph representation are analyzed to derive a rule-based method for root node selection. However, this approach would require medical knowledge about the anatomical structure of the coronary tree.

4.3. Conclusion

In this project, we developed two promising approaches using machine learning in medical image processing.

The first approach focuses on Automated Coronary Artery Segmentation, adapting the UNETR model, which utilizes attention heads in the encoding process [5], to suit the specific requirements of our task. We experimented with optimizing architectural choices, hyperparameters, and post-processing techniques to enhance segmentation performance.

The second approach involves Coronary Tree Graph Representation, using a pipeline to algorithmically convert the predicted segmentation mask from Automated Coronary Artery Segmentation into a simplified graph representation. This is achieved by removing unnecessary nodes, maintaining graph size, and extracting structural characteristics. The resulting graph representation provides valuable metrics for analyzing the coronary tree.

Overall, this project demonstrates the potential of machine learning and graph-based methods for coronary artery segmentation and analysis. While our approaches show promising results, challenges remain, particularly in computational efficiency, model optimization, and medical data processing. With further refinement, these methods could help develop more efficient and reproducible tools for coronary artery assessment.

References

[1] A. K. Malakar, D. Choudhury, B. Halder, P. Paul, A. Uddin, S. Chakraborty, A review on coronary artery disease, its risk factors, and therapeutics, Journal of cellular physiology 234 (2019) 16812–16823.

[2] R. D. Shahjehan, S. Sharma, B. S. Bhutta, Coronary artery disease, in: StatPearls [Internet], StatPearls Publishing, 2024.

[3] D. Veiga-Canuto, L. Cerdà-Alberich, C. Sangüesa Nebot, B. Martínez de Las Heras, U. Pötschger, M. Gabelloni, J. Carot Sierra, S. Taschner-Mandl, V. Düster, A. Cañete, et al., Comparative multicentric evaluation of inter-observer variability in manual and automatic segmentation of neuroblastic tumors in magnetic resonance images. cancers. 2022; 14: 3648, DOI: https://doi. org/10.3390/cancers14153648 (2022).

[4] J. Ma, Y. He, F. Li, L. Han, C. You, B. Wang, Segment anything in medical images, Nature Communications 15 (2024) 654.

[5] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. R. Roth, D. Xu, Unetr: Transformers for 3d medical image segmentation, in: Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2022, pp. 574–584.

[6] O. Ronneberger, P. Fischer, T. Brox, 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, Springer, 2015, pp. 234–241.

[7] S. L. Smith, P.-J. Kindermans, C. Ying, Q. V. Le, Don't decay the learning rate, increase the batch size, arXiv preprint arXiv:1711.00489 (2017).

[8] F. Cademartiri, E. Maffei, A. A. Palumbo, R. Malagò, L. La Grutta, W. B. Meiijboom, A. Aldrovandi, M. Fusaro, L. Vignali, A. Menozzi, et al., Influence

of intra-coronary enhancement on diagnostic accuracy with 64-slice ct coronary angiography, European radiology 18 (2008) 576–583.

[9] B. Liu, J. Dolz, A. Galdran, R. Kobbi, I. B. Ayed, Do we really need dice? the hidden region-size biases of segmentation losses, Medical Image Analysis 91 (2024) 103015.

[10] S. Shit, J. C. Paetzold, A. Sekuboyina, I. Ezhov, A. Unger, A. Zhylka, J. P. Pluim, U. Bauer, B. H. Menze, cldice-a novel topology-preserving loss function for tubular structure segmentation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 16560–16569.

[11] T.-C. Lee, R. L. Kashyap, C.-N. Chu, Building skeleton models via 3-d medial surface axis thinning algorithms, CVGIP: graphical models and image processing 56 (1994) 462–478.

[12] M. A. Padalino, N. Franchetti, M. Hazekamp, V. Sojak, T. Carrel, A. Frigiola, M. Lo Rito, J. Horer, R. Roussin, J. Cleuziou, et al., Surgery for anomalous aortic origin of coronary arteries: a multicentre study from the european congenital heart surgeons association, European journal of cardio-thoracic surgery 56 (2019) 696–703.

[13] J. Reichold, M. Stampanoni, A. L. Keller, A. Buck, P. Jenny, B. Weber, Vascular graph model to simulate the cerebral blood flow in realistic vascular networks, Journal of Cerebral Blood Flow & Metabolism 29 (2009) 1429–1443.

A. Appendix

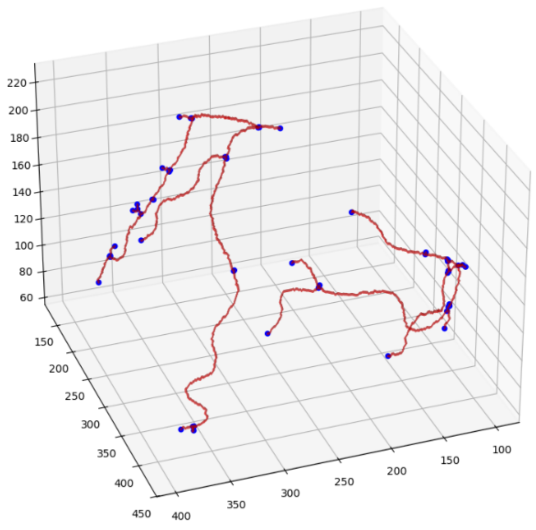
Figure 21: Skeleton and nodes representation before merging close nodes (Number of nodes = 58, Number of edges = 66)

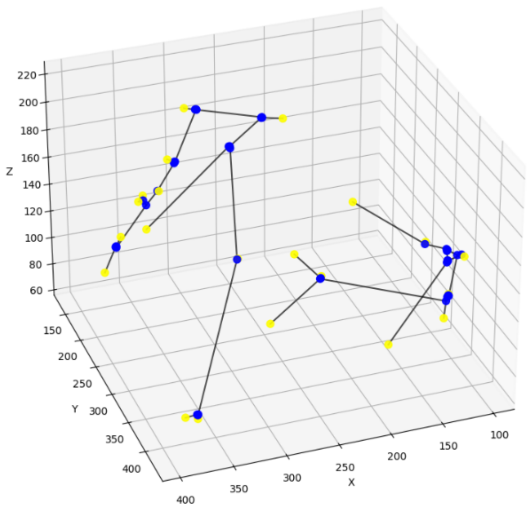

Figure 22: Graph representation before merging close nodes (Number of nodes = 58, Number of edges = 66)

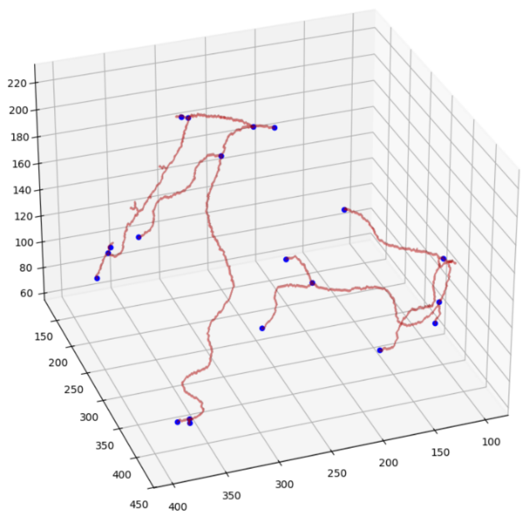

Figure 23: Skeleton and nodes representation after merging close nodes with $d = 5$ (Number of nodes = 20, Number of edges = 18)

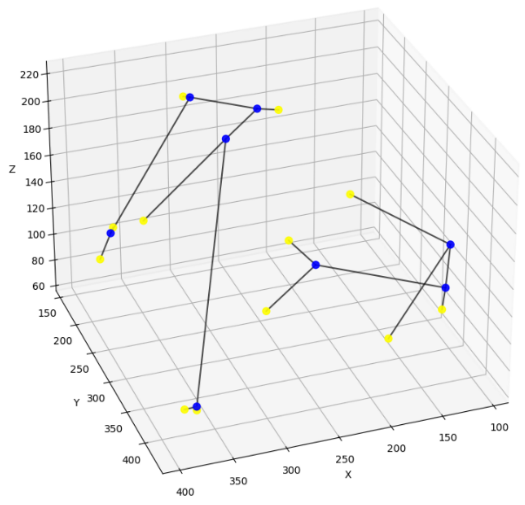

Figure 24: Graph representation after merging close nodes with $d = 5$ (Number of nodes = 20, Number of edges = 18)

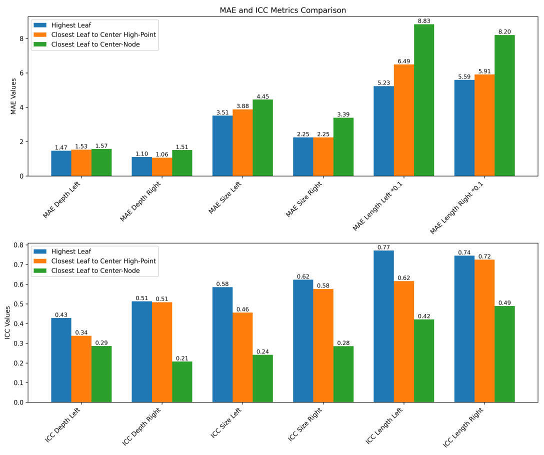

Figure 25: Comparison of root node selection methods