3D CT Proximal Femora Segmentation Using Optimized nnU-Net

Interdisciplinary Project Report

Jeremias Lang
Student ID: 11706731

Main Supervisor: Univ.Ass. Dipl.-Ing. Dr.techn. Sebastian Bachmann
Co-Supervisor: Univ.Lektor Dr.techn. Alexander Pacha MSc

September 22, 2024

Contents

1 Abstract **1**

2 Motivation **1**

3 State of the Art **2**

4 Problem Statement & Research Questions **3**
4.1 Data 4

5 Methodology **4**
5.1 nnU-Net 4
5.1.1 Architectural Details 5
5.1.2 Usage 6
5.2 Experimental Setup 6
5.3 Performance Metrics 8

6 Evaluation **9**
6.1 Parameter Search 9
6.2 Test Set Evaluation 14
6.2.1 Manually Corrected Labels 14
6.2.2 Original Labels 17

7 Conclusion **21**

1 Abstract

This project implements the optimization and evaluation of an nnU-Net model for femur 3D Computed Tomography (CT) segmentation, an important step for applications like biomechanical analysis and medical planning. The study compares both 2D and 3D configurations of the model, evaluating their performance using metrics such as Dice score, Sensitivity, Specificity, and Average Surface Distance (ASD) as well as visualizations. The best performing models, "2D, small, ResEnc" and "3D, small," were trained on the full dataset for 1000 epochs and evaluated on a test split. The 3D model consistently outperformed the 2D model in all metrics and was able to achieve similar results compared to the performance scores of Yosibash et al. [1] and Deng et al. [2], who segmented similar data, while having access to fewer images. Inference times and VRAM usage were also examined, with the 3D model being faster but requiring more VRAM. The project concludes by highlighting the potential to further improve segmentation performance through more consistent ground truth data. The code for this project is provided in a GitHub repository [3].

2 Motivation

Accurate and efficient segmentation of bones from clinical CT images is the first important step in the development of Finite Element (FE) models. These models are critical

for various different applications ranging from medical planning to biomechanical analysis. However, the process is often hindered by the need for manual correction, making it time-consuming and prone to human error. Traditional methods for automatic medical image segmentation, such as the algorithm "fill" in the paper of Pahr et al. [4], usually are based on global thresholding and morphological operations. Global thresholding has the disadvantage of struggling to accurately replicate local features of the image, which is particularly problematic for complex bone structures. Specifically for the "fill" method, shapes with openness greater than 45 degrees, such as "C" shaped objects, are not accurately captured. This can lead to incomplete or inaccurate representations of the bone structure, which then have to be manually corrected. An example of a segmentation given by the "fill" algorithm is visualized in Figure1. In contrast, deep learning in the form of Convolutional Neural Networks (CNNs), which are generally recognized for their ability to capture local features in images, promise great improvements. Unlike traditional methods, CNNs could better handle intensity inhomogeneities, noise, and artifacts common in CT scans [5]. This is particularly relevant for FE modeling, where precise segmentation masking is essential.

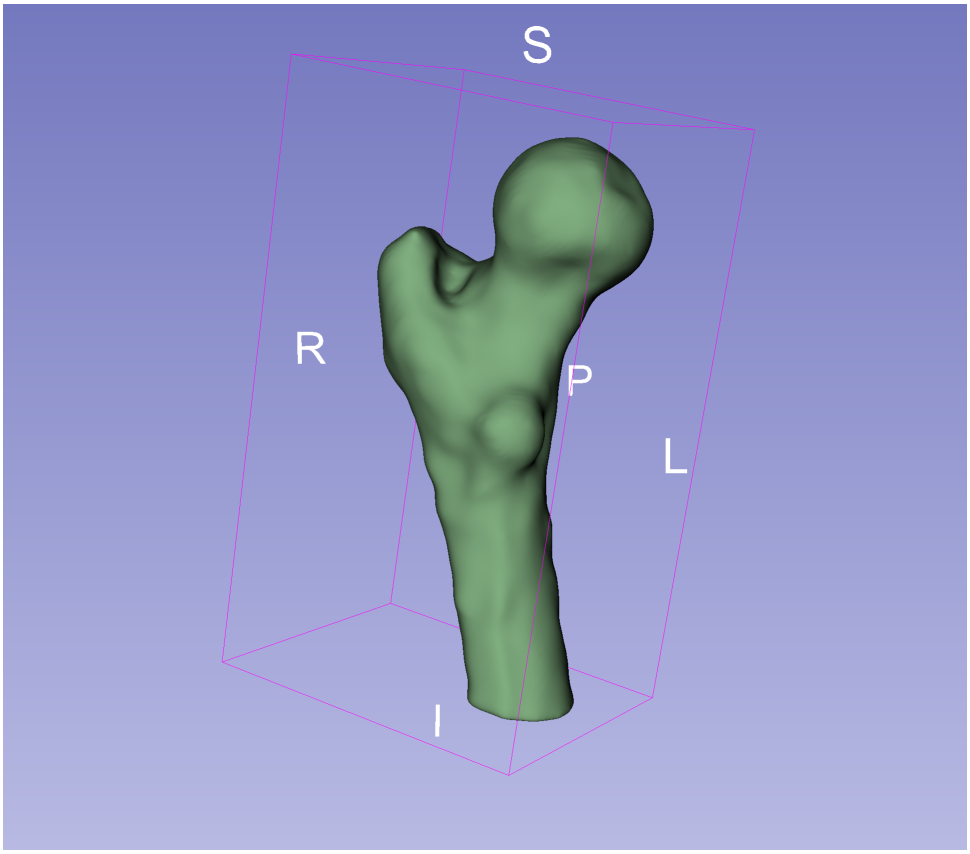

Figure 1: Example of a femur CT segmentation created by the "fill" algorithm.

3 State of the Art

In medical imaging, segmentation faces great challenges not just due to technical factors such as varying image quality from different scanners but also due to the natural and pathological variability in anatomical structures. Human anatomy differs significantly across individuals, and these variations are even further increased by potential patholog-

ical differences. This project focuses on the natural variability in anatomical features, excluding pathological samples to avoid additional complexities. The goal is to develop robust segmentation methods that can adapt to the natural variability of the bone structure.
The currently used method for the outer masking of the proximal femora is the before mentioned "fill" [4] algorithm. The state of the art in machine learning for medical image segmentation can be summarized as based on U-Net models [6], transformers, or a combination of both. While recent research suggests slightly superior performance of big, computationally heavy transformer models like "MedSAM" [7] and "Swin UNETR" [8], their performance seems to only barely surpass one of the older U-Net based models "nnU-Net" [9] in most data sets (in some nnU-Net wins), as shown in a recent paper of the 3D "TransU-Net" [10]. Additionally, there is recent work claiming that several new architectural approaches suffer from bias in their evaluation [11]. Further, the ex vivo images of proximal femora seem to be data where U-net based models perform well [1]. A reason for this might be the high image quality and standardized scan parameters typical of ex vivo imaging. Such images usually have minimal soft tissue interference, providing clearer data compared to clinical CT scans. Even though nnU-Net is already an older approach, it is still being used to compete in recent segmentation challenges[12] and was therefore chosen as the model architecture for this work.
Another important decision in model architecture is choosing between a 2D slice-wise and a 3D approach. The 2D method, while lacking spatial information, provides more samples and is computationally simpler. In contrast, the 3D approach, though computationally more demanding with fewer samples, incorporates valuable spatial information of the full image. Although there have been reviews [13, 14] about this trade-off, the decision is not straightforward, as the effectiveness of these approaches is data-dependent, and it is currently not clear which method is inherently superior for bone segmentation in CT images. As such, both the 2D and 3D approach will be implemented and compared with each other.
While not utilizing the same data, the results from [2] and [1] will be used as comparable baselines.

4 Problem Statement & Research Questions

This project involves the implementation, optimization, and evaluation of an nnU-Net model to predict segmentation masks for a specific femur CT dataset, as detailed in Section 4.1. The effectiveness of the nnU-Net model for this dataset will be assessed by answering the following research questions:

1. How effectively does the nnU-Net model approximate the ground truth defined by the manually corrected images from the "fill" method? This will be tested by comparing the segmentation results of nnU-Net with the ground truth using the Dice score as the primary metric for evaluation.

2. What are the performance differences between the best 2D and 3D versions of nnU-Net in segmenting proximal femora in ex vivo CT images? Both approaches will be implemented and their Dice scores as well as their Sensitivity, Specificity and ASD compared.

3. How is the performance of the default nnU-Net affected by optimizing for selected hyperparameters? The effectiveness of the configurations will be evaluated based on improvements in model performance on a separate validation set, measured using the Dice score.

4. How does the choice of model (2D vs. 3D) affect the training time and resource utilization? The duration and resources will be measured for each segmentation training and inference process.

5. What is the impact of training the model on the original dataset, which was segmented using the "fill" method, vs. training the model on the dataset that has undergone manual correction? This will be assessed by comparing the model's Dice score, Sensitivity and Specificity and ASD.

6. How does the performance of the nnU-Net model hold up to the expected results for femur CT image segmentation? This will be quantitatively assessed by comparing the results with those obtained in [2] and [1].

4.1 Data

The dataset for this study originates from the paper of Dall'Ara et al. [15]. Available are 36 paired femora (72 scans) with an extra set of 4 pairs (8 scans) of damaged, deformed or problematic femora, which were not included in the original study. The specimens, from 17 male and 19 female donors aged 76 $\pm$ 12 years, were prepared by removing soft tissues and embedding the distal portions in Polyurethane to stabilize the bone during the scanning process, ensuring the images are clear and undistorted. They were scanned alongside a calibration phantom in a clinical Quantitative Computed Tomography (QCT), with specific settings to optimize image quality and accuracy for bone density measurements. The dimensions of the images span from a minimum of [153, 69, 94] to a maximum [199, 93, 133] in each respective dimension with isotropic spacing of 1 mm.

5 Methodology

This section first discusses the architectural details of the nnU-Net, followed by an explanation of how exactly the experiments were planned and conducted.

5.1 nnU-Net

The U-Net architecture was a breakthrough in the field of biomedical image segmentation and classification, improving the accuracy of pixel-wise classification problems. It introduced a fully convolutional network with a symmetric encoder-decoder structure. The encoder consists of a type of CNN, more specifically repeated convolutional layers followed by activation functions and downsampling operations, which reduce spatial resolution while gaining more context. The decoder roughly mirrors the encoder structure but performs upsampling operations to recover to the full resolution, enabling precise localization. The use of skip connections between layers on the same level in the encoder and decoder further improves localization by merging high-resolution features from the

encoder with the upsampled features in the decoder directly without them being downsampled first [6].

The nnU-Net (no new U-Net) framework is a self-adapting system built on a set of three basic U-Net models. The automated configuration spans from preprocessing to postprocessing. Its architecture is adapted based on certain heuristics dependent on the "fingerprint" of the data, capturing some of its characteristics. The framework includes three basis models: a 2D and 3D U-Net for full-resolution segmentation and a U-Net "cascade" model that first processes images with lower resolution and then on full-resolution after upsampling.

5.1.1 Architectural Details

The nnU-Net framework consists of three types of design rules: fixed, rule-based and empirical parameters. The following summarized information and more technical details can be found in the supplementary material provided by the authors of the nnU-Net [16].

Fixed Parameters The fixed parameters define the rigid core structure of the nnU-Net models. These parameters were empirically found to work well for segmentation tasks in general. The architecture consists of convolutional layers, instance normalization, and Leaky ReLU activation functions always arranged in the specific sequence: convolution, instance normalization, and Leaky ReLU. Each resolution stage in the encoder and decoder has two computational blocks. Downsampling is performed by using strided convolutions, upsampling is performed by using transposed convolutions.
During preprocessing, all data is initially cropped to the region containing nonzero values. To ensure consistent spatial semantics, all images are resampled to the median voxel spacing of their respective dataset. For CT scans, intensity values within the segmentation masks of the training dataset are collected, and normalization is performed by clipping to the [0.5, 99.5] percentiles, followed by z-score normalization based on the mean and standard deviation of all collected intensity values. If cropping significantly reduces the size of a dataset, normalization is carried out only within the mask of nonzero elements, with all values outside the mask set to zero.
There exist three predefined models: 2D U-Net, 3D full resolution U-Net, and 3D U-Net cascade. The 2D U-Net processes full resolution data and seems to be particularly effective for anisotropic datasets. The 3D full resolution U-Net also handles full resolution data, but its patch size is limited by larger GPU memory requirements and seems to perform better on smaller datasets. The 3D U-Net cascade model first processes images with lower resolution and then on full-resolution after upsampling. This approach often demonstrated superior performance for very large images. However, in this project, the images are small enough that this third approach is unnecessary and will not be included in the experiments.
All models are trained for a fixed length of 1000 epochs, with each epoch consisting of 250 iterations, relative to the batch size. The training process uses stochastic gradient descent with an initial learning rate of 0.01 and a Nesterov momentum of 0.99. The learning rate is progressively reduced using the 'polyLR' schedule, which decreases almost linearly to zero over the course of training. Data augmentation is performed in parallel to generate a continuous stream of augmented images and includes random rotations, random scaling,

random elastic deformations, gamma correction augmentation and mirroring. The Dice loss is combined with the cross-entropy loss to improve training stability and segmentation accuracy.
By default, inference is done using an ensemble of models trained on the validation splits within a 5-fold cross-validation. The inference is conducted on a patch-wise basis, with overlapping predictions made to prevent stitching artifacts by a distance of half the patch size. Predictions near the borders are performed using Gaussian importance weighting for softmax aggregation.

Rule-Based Parameters The rule-based parameters are automatically configured for each dataset based on its specific characteristics. These parameters define the network architecture by adapting to the size and spacing of input patches during training. The downsampling process continues until the feature maps are relatively small in order to obtain enough context. The network's depth is dependent on the input patch size, where the number of convolutional layers is exactly $5 \cdot k + 2$, with k being the number of downsampling operations. Additional loss functions are applied to all but the two lowest resolutions of the decoder to maintain gradient flow throughout the network.
The input patch size is configured to be as large as possible while still allowing a batch size of 2. The aspect ratio of the patch size is determined by the median shape of the resampled training cases. The batch size is kept at a minimum of 2 to ensure robust optimization, but if some GPU memory is still left unused after defining the patch size, the batch size is increased as much as possible.
The median spacing of the training cases is used as the default for isotropic data, with resampling done using third-order splines for data and linear interpolation for one-hot encoded segmentation maps.

Empirical Parameters The performance of 2D versus 3D models cannot be predicted beforehand, so the model selection must be made after training and evaluating all the different architectures on the validation split. Postprocessing in the form of connected component analysis might improve prediction accuracy by keeping only the largest component, but its effectiveness has to be confirmed through testing on the validation split.

5.1.2 Usage

The nnU-Net code is publicly accessible on GitHub, providing instructions for installation and usage [17]. The repository includes detailed guidance on how to preprocess data, train models, and make predictions using specific terminal commands. In addition to directly modifying the Python files, the repository supports indirect parameter adjustments through the nnU-Net "plans" JSON files, allowing one to more easily change certain parameters of the architecture and training process.

5.2 Experimental Setup

Firstly, in order to obtain ground truth segmentations as accurate as possible for this project, all 72 3D CT images processed by the "fill" algorithm are manually corrected by the author to the best of his knowledge using the program "3D Slicer" [18]. Before

training, 10 images are set aside as the test dataset. The "nnU-Net" models are then trained on the remaining CT images and made accessible via Python Jupyter notebooks. For both the 2D and 3D models, this includes additional hyperparameter tuning as recommended in the paper by Isensee et al. [12], in particular a complete grid-search on the usage of the residual encoder, varying batch size and a different data augmentation preset "DA5". While initially planned in the proposal, different patch sizes were not tested as increasing the batch size did not result in improved performance. The relatively small spatial resolution of the images allowed the patch size to cover the whole image in the default configuration, and reducing it to increase the batch size further therefore was deemed unnecessary. The batch sizes were increased until they barely still fit in the VRAM of the GPU, with the exact values depicted in Table 1.

Parameter	**Values**
Dimensional Structure	2D, 3D
Encoder Structure	Default, ResEnc
Batch Size (2D Default)	Small (372), Medium (744), Large (1116)
Batch Size (2D ResEnc)	Small (377), Medium (558), Large (744)
Batch Size (3D Default)	Small (2), Medium (4), Large (6)
Batch Size (3D ResEnc)	Small (3), Large (4)
Data Augmentation	Default, DA5

Table 1: Parameter Ranges for nnU-Net Training. ResEnc stands for the usage of the medium residual encoder preset "nnU-Net ResEnc M", which is the largest that would still fit into 16GB VRAM. DA5 is a more aggressive data augmentation preset.

Each configuration is trained only on a single fold of the default five-fold cross-validation for 100 epochs to reduce training time, making it feasible to train all possible combinations as each configuration trained for about 4 hours on average. The best 2D and 3D model configurations are then selected based on the mean validation Dice score, computed on the validation fold used during the parameter search. The chosen best 2D and 3D configurations are then trained on the full training data split for the default 1000 epochs. To improve inference time, only a single model is used for the final configuration, rather than the default ensemble of the five-fold cross-validation models. After training, these models are evaluated on the test set. The primary metric for this evaluation is the Dice score. However, also the sensitivity, specificity, and ASD as well as visualizations of the prediction errors are implemented for a more comprehensive evaluation.
To further analyze the models' robustness, their performance is also tested on the original images without manual corrections to determine how the ground truth labeling provided by the "fill" method impacts the nnU-Net's training. Finally, the performance of the best model is compared against the performance scores reported by Deng et al. [2] and Yosibash et al. [1].
In addition, the training and inference times, along with GPU resource utilization, were measured. Details for the setup that was used for all experiments are depicted in Table 2.

Component	Specification
CPU	12 × AMD Ryzen 5 3600 6-Core Processor
RAM	15.6 GiB
GPU	NVIDIA GeForce RTX 4060 Ti/PCIe/SSE2
VRAM	16 GB
OS	Fedora 40

Table 2: Specifications of the system used for training and inference.

In order to ensure reproducibility of the results, all code is made public in a GitHub repository , complete with comments and documentation. Additionally, the exact versions of all used software and libraries are specified, and a guide for setup and execution of the models is provided.
The workflow is depicted graphically in Figure 2.

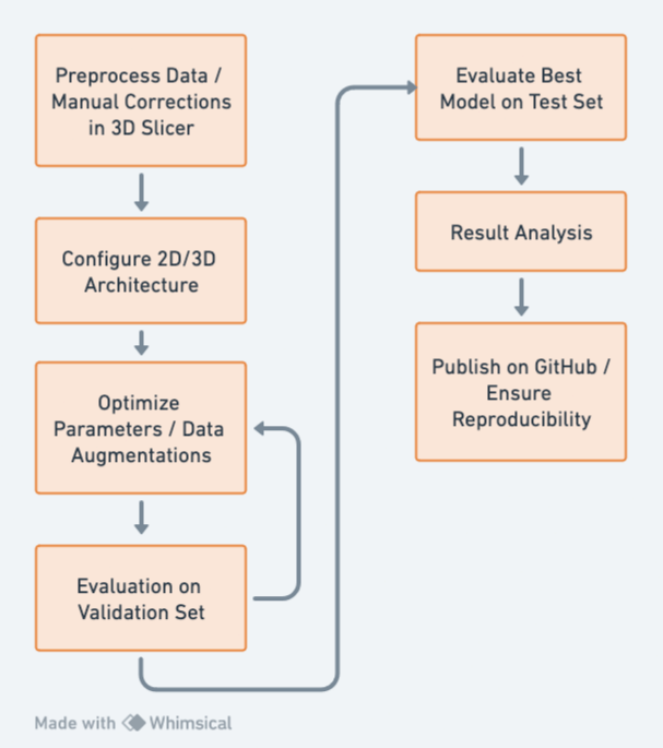

Figure 2: Graphic giving an overview of the workflow for this project.

5.3 Performance Metrics

The performance metrics used to evaluate the segmentation performance of the models are: Dice Coefficient, Sensitivity, Specificity, and Average Surface Distance (ASD).

The Dice Coefficient is the primary metric for assessing segmentation performance. It measures the overlap between the predicted segmentation and the ground truth. The value ranges from 0 to 1, where a Dice score of 1 represents perfect agreement between the prediction and ground truth. It provides a balanced evaluation of precision and recall. The definition is given as:

$$\text{Dice} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

where TP are true positives, FP are false positives and FN are false negatives.

Sensitivity evaluates the ability of the model to correctly identify positive instances (here, pixels that are considered femur). The formula is:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

A high sensitivity indicates that the model successfully captures most of the bone structure, but it does not penalize false positives, meaning that the model might predict extra bone regions.

Specificity measures the ability of the model to identify negative instances (here, pixels that are background). The formula is:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

High specificity indicates that the model is effective in excluding non-bone pixels from the segmentation. It acts as a measure for avoiding over-segmentation of the bone.

The ASD is defined as the Average Euclidean Distance between the surface of the predicted segmentation and the surface of the ground truth segmentation. Lower ASD values mean better contour matching, which is essential for precise localization of the bone boundaries. The formula is:

$$\text{ASD} = \frac{1}{|S_p| + |S_t|} \left(\sum_{p \in S_p} d(p, S_t) + \sum_{t \in S_t} d(t, S_p) \right)$$

where S_p and S_t are the sets of surface points from the predicted and true segmentations, respectively, and $d(p, S_t)$ represents the euclidean (shortest) distance from a point p to the surface S_t.

6 Evaluation

This section analyzes the parameter search and the final performance of the best selected 2D and 3D models on the test images, comparing results using both the manually corrected and uncorrected labels generated by the "fill" algorithm.

6.1 Parameter Search

Training each configuration for only 100 epochs during the parameter search appeared to be enough, as the performance metrics during training seemed to converge quickly, as shown in Figure 3.

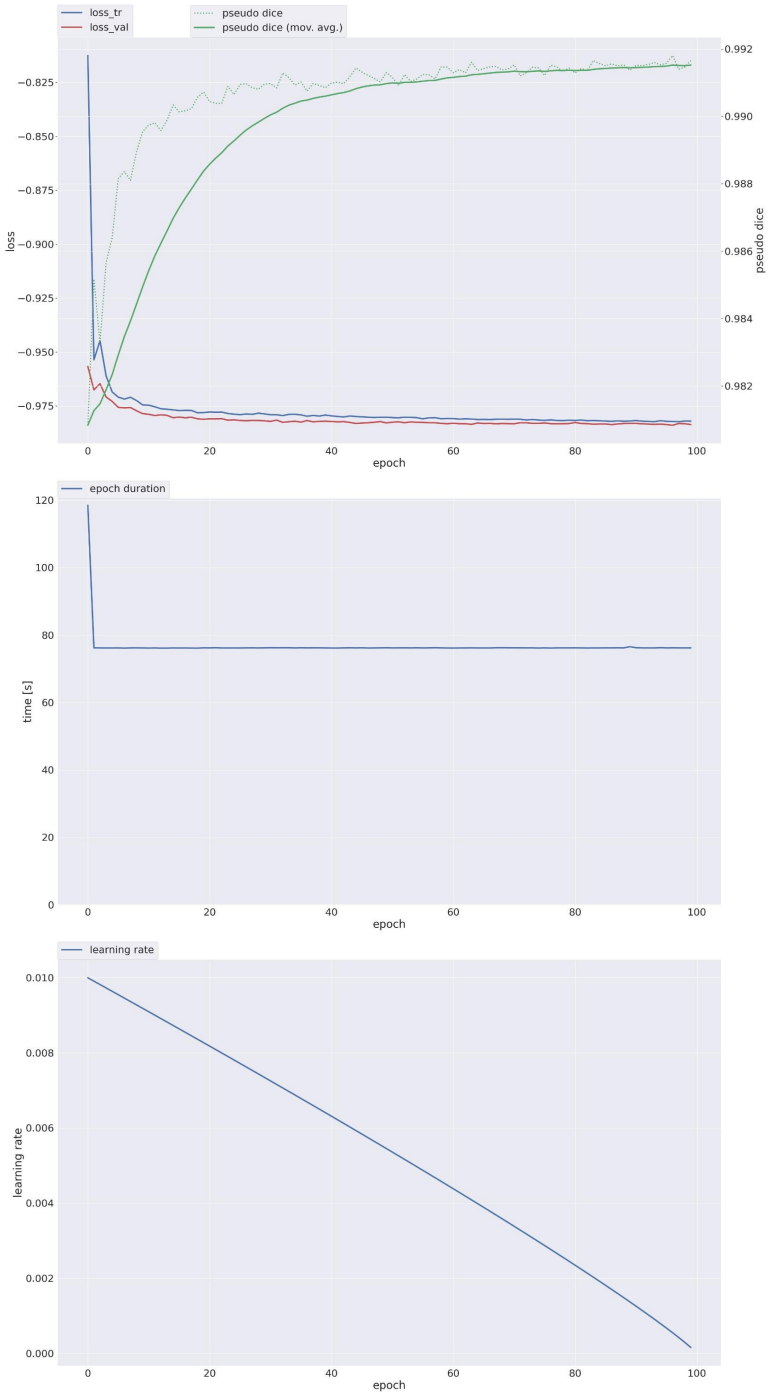

Figure 3: Top: Training progress for the "3D, small" model over 100 epochs on one fold of the 5-fold cross-validation. The loss represents the sum of the cross-entropy score and the negative Dice score, while the pseudo-Dice score calculates the Dice score using a batch as a pseudo-volume for faster computation during training. Middle: Training time across all epochs. Bottom: Learning rate across all epochs.

The Dice scores resulting from the parameter search to optimize the nnU-Net structure are shown in Table 3. The default performance of both the 2D and 3D networks was already high, with Dice scores exceeding 0.99. Increasing the batch size did not lead to any improvements across all configurations. Similarly, the use of the DA5 augmentation preset showed no significant effect on performance.

All 3D configurations slightly but consistently outperformed the 2D configurations, as illustrated in Figure 4. Among the 2D models, the best performance was achieved by the "2D, small, ResEnc" configuration. The best performing model overall was the default "3D, small" configuration. These two models were selected for further evaluation, undergoing training on the full dataset for 1000 epochs using both the manually corrected and original labels, with results evaluated on the test dataset in the next section.

The lack of significant improvement from the parameter search may be due to the clarity of the bone images and the relative simplicity of the segmentation task, allowing the default nnU-Net settings to already approach near-optimal performance. Additionally, the ground truth labels were created manually by a non-expert, introducing human error. There were many ambiguous regions during the manual correction of the labels where it was unclear whether certain pixels should be labeled as bone or background, leading to inconsistencies. Given these labeling inaccuracies, achieving a perfect Dice score is inherently impossible. Therefore, the 0.9917 Dice score could already represent a near-optimal outcome, constrained by the quality of the ground truth labels rather than the model structure.

Configuration	**Validation Dice Score**
2D, small	0.99021
2D, medium	0.99017
2D, large	0.99015
3D, small	**0.99172**
3D, medium	0.99153
3D, large	0.99167
2D, small, ResEnc	0.99052
2D, medium, ResEnc	0.99050
2D, large, ResEnc	0.99049
3D, small, ResEnc	0.99162
3D, large, ResEnc	0.99171
2D, small, DA5	0.99013
2D, medium, DA5	0.99004
2D, large, DA5	0.99000
3D, small, DA5	0.99124
3D, medium, DA5	0.99137
3D, large, DA5	0.99148
2D, small, DA5, ResEnc	0.99046
2D, medium, DA5, ResEnc	0.99038
2D, large, DA5, ResEnc	0.99044
3D, small, DA5, ResEnc	0.99127
3D, large, DA5, ResEnc	0.99143

Table 3: Validation Dice Scores for Different Configurations using the corrected labels. The best score is highlighted in bold.

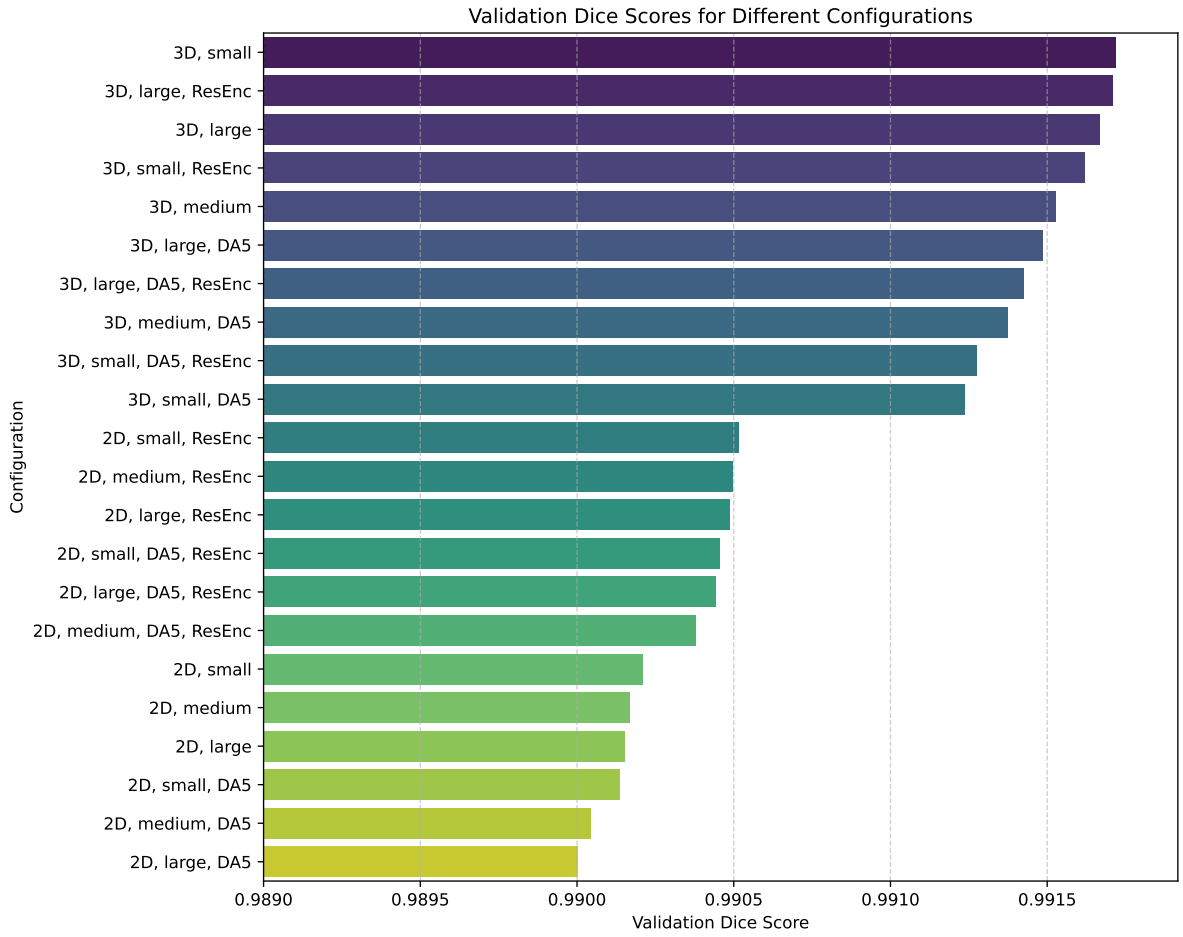

Figure 4: Validation Dice Scores for Different Configurations using the corrected labels.

The VRAM and RAM usage for each configuration during training is shown in Figure 5. As expected, the largest increase in VRAM usage was caused by increasing the batch size and the use of the residual encoder. It is important to note that the "large, ResEnc" and "large, default" configurations do not correspond to the same batch sizes. The exact batch sizes associated with each configuration name are listed in Table 1. CPU RAM usage showed no clear correlation with different settings or VRAM usage. The DA5 data augmentation method had no significant impact on resource usage and was therefore omitted from testing. Additionally, the choice between 3D and 2D models did not appear to be linked to higher resource usage.
The configuration with the highest VRAM requirement was "2D, large," which used approximately 14.2 GiB of VRAM. On the other hand, the best-performing model, "3D, small," required only 4.7 GiB of VRAM, showing that the best performance was obtained with relatively low resources.

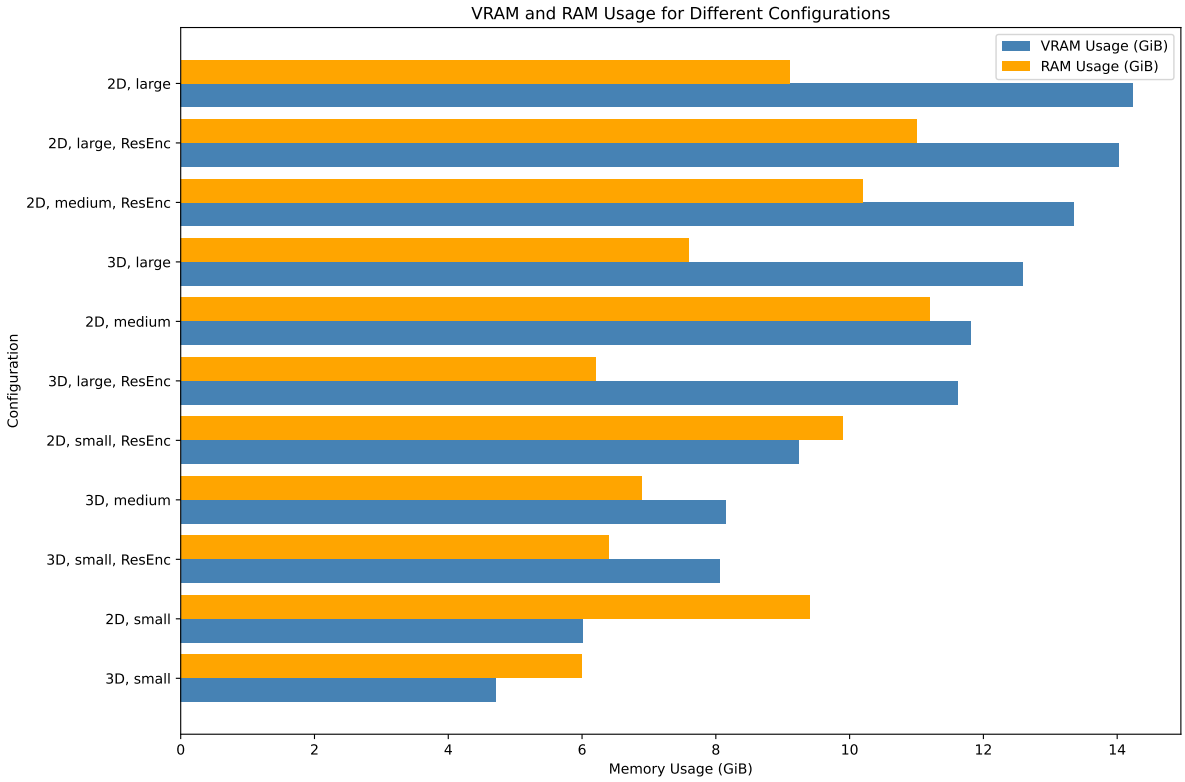

Figure 5: VRAM and RAM Usage for Different Configurations.

The training times for each configuration, as shown in Figure 6, mirror roughly the VRAM usage. Increasing the batch size or using the residual encoder significantly increased the training time. The longest training duration was observed with the "2D, large, ResEnc" configuration, which took approximately 8 hours and 30 minutes. On the other hand, the fastest training time was achieved by the best-performing model, "3D, small," which completed training in approximately 2 hours. The DA5 data augmentation had no significant impact on training time and was therefore omitted from testing again.

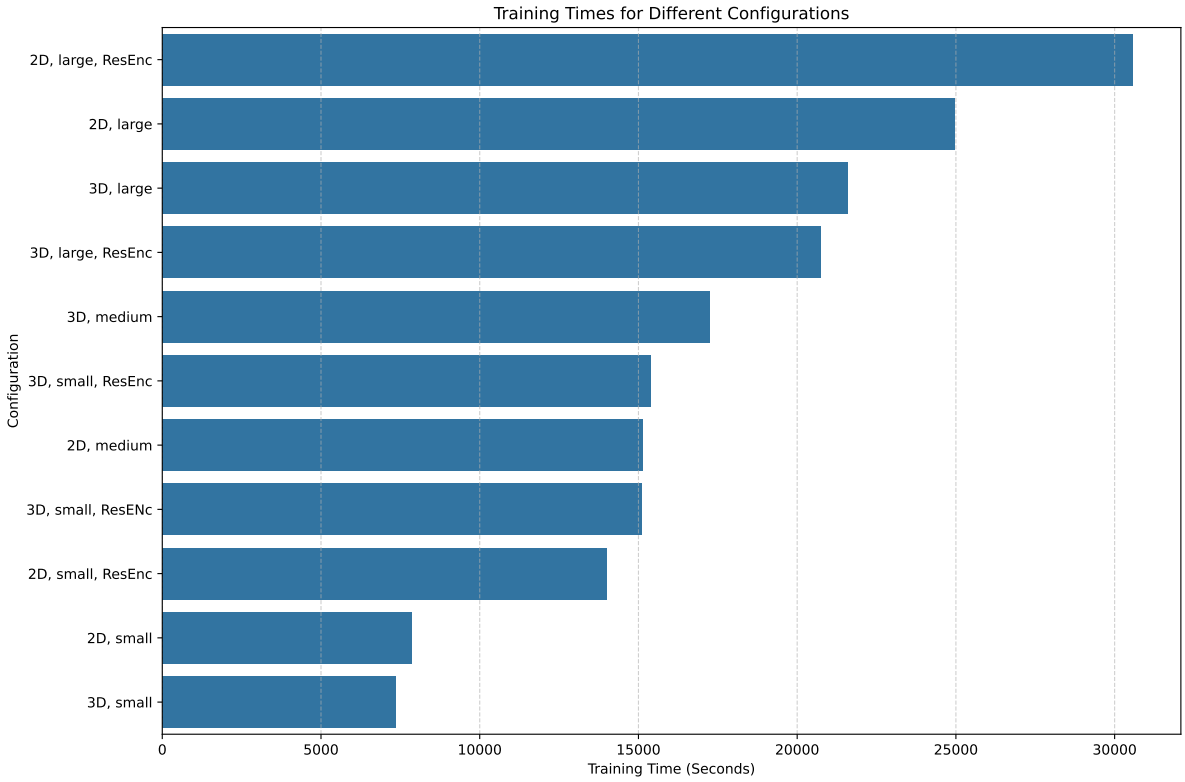

Figure 6: Training times for all configurations for 100 epochs on one split of the 5-fold cross validation.

6.2 Test Set Evaluation

6.2.1 Manually Corrected Labels

After training the chosen "2D, small, ResEnc" and "3D, small" models on the full training dataset for 1000 epochs, they were used to predict the 10 images in the test split, using the manually corrected labels. These pretrained models are available for installation in the repository [3]. The performance metrics obtained from the manually corrected ground truth labels, along with comparable baseline performance scores for femur CT segmentation, are shown in Table 4.

As one can see in the results, the 3D model outperforms the 2D model across all metrics, with both achieving relatively high Dice scores above 0.97. These differences are visualized more clearly in Figure 7. When compared to the results of Deng et al. [2] and Yosibash et al. [1], the performance of the models lies in between, with Yosibash et al. achieving a slightly higher Dice score. One reason for this might be their larger dataset of 221 images, compared to the 72 images used in this study. Additionally, as mentioned before, the ground truth labels in this project contain inconsistencies, which might also contribute to the slightly lower performance.

The ASD of the models was surprisingly better than both baselines, with especially the 3D model showing superior performance. Both the sensitivity and specificity of the models follow a similar trend to the baselines: while the specificity is closer to optimal, the sensitivity is slightly but consistently lower. This suggests that the models are better at correctly identifying non-bone regions (high specificity) than correctly identifying all

bone regions (lower sensitivity). This is not that surprising, as there is generally more background than bone in each image.

Model	Dice	Sensitivity	Specificity	ASD
2D	0.9879	0.9895	0.9979	0.1424
3D	0.9895	0.9914	0.9981	0.1246
Deng et al. [2]	0.9782	0.9959	0.9989	0.1657
Yosibash et al. [1]	0.9913	0.9892	0.9998	0.1526

Table 4: Mean performance metrics for the 2D and 3D models on the test set images using manually corrected labels, together with comparable baseline performance metrics. The scores from Deng et al. [2] represent the performance of the periosteal surface segmentation, while the scores from Yosibash et al. [1] represent the femur segmentation performance for a 150mm length.

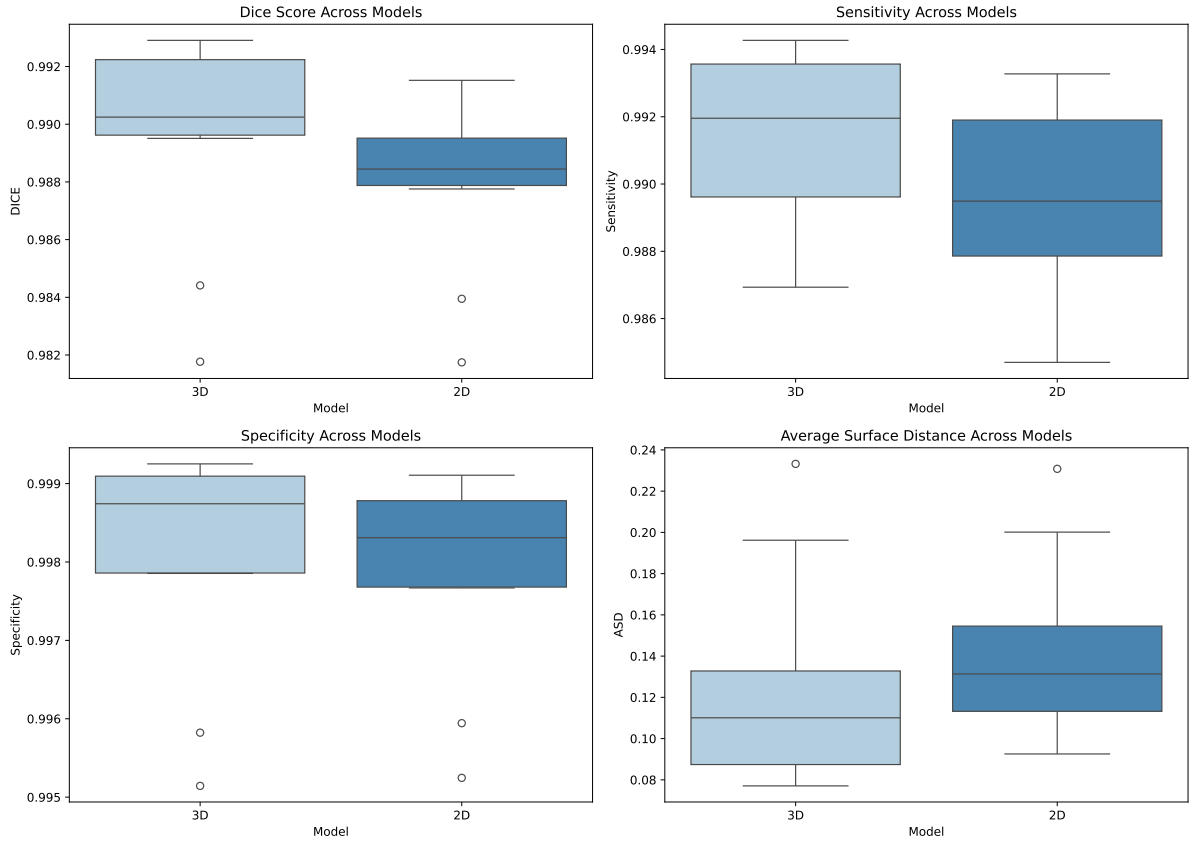

Figure 7: Boxplots for the mean performance metrics for the 2D and 3D models on the images in the test set using manually corrected labels.

The inference times and VRAM usage are shown in Table 5. The 3D model predicts faster, as expected, since it does not need to process as many slices in series. However, it required significantly higher VRAM usage.
To get a more detailed look of the segmentations and their differences from the ground truth labels, the provided Jupyter notebooks allow for the generation of error overlay

maps. These notebooks include interactive plots where one can use sliders to search through the predicted images and choose the desired model for comparison. An example of such an error map is shown in Figure 8. As the errors are only located at the outer parts of the bones, a more visually clear representation might be the contour visualization, as shown in Figure 9.

Model	**Inference time per image (seconds)**	**VRAM usage (MiB)**
3D	1.68	2000
2D	6.00	408

Table 5: Inference time and VRAM usage for 2D and 3D models.

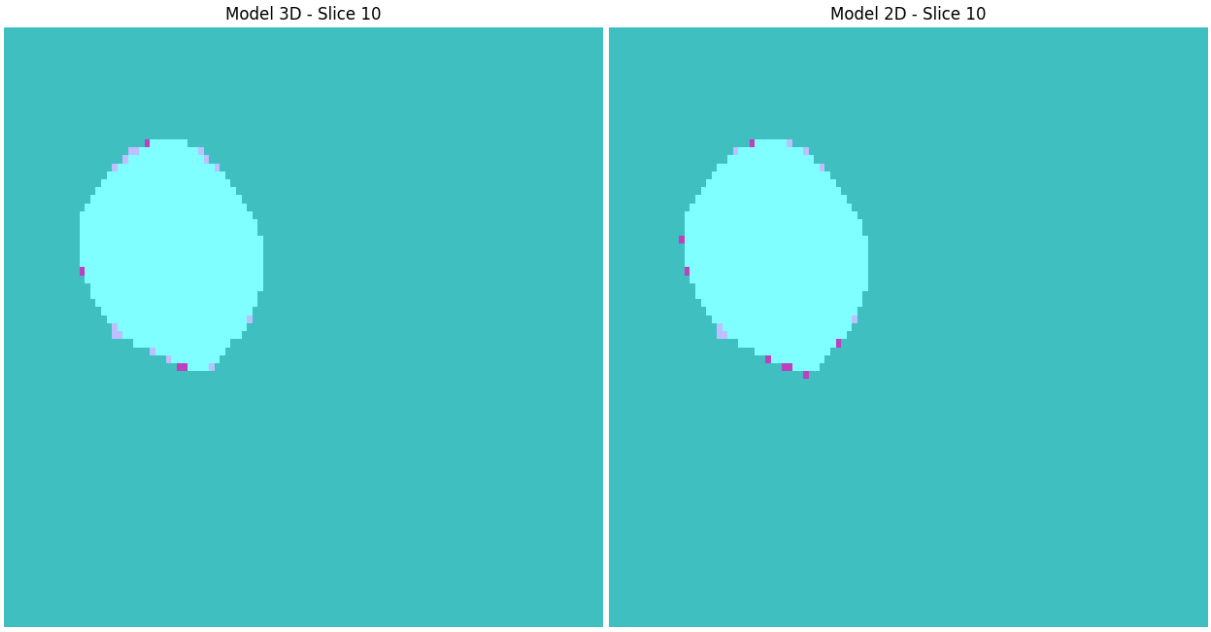

Figure 8: Error overlay visualization of the 2D and 3D model segmentations compared to the ground truth for image "F7L_fall.nrrd". Light blue indicates correctly predicted bone, while dark blue represents correctly predicted background. Dark violet represents pixels labeled as bone in the prediction but are background in the ground truth labels, and light violet indicates pixels labeled as background in the prediction but are actually bone in the ground truth labels.

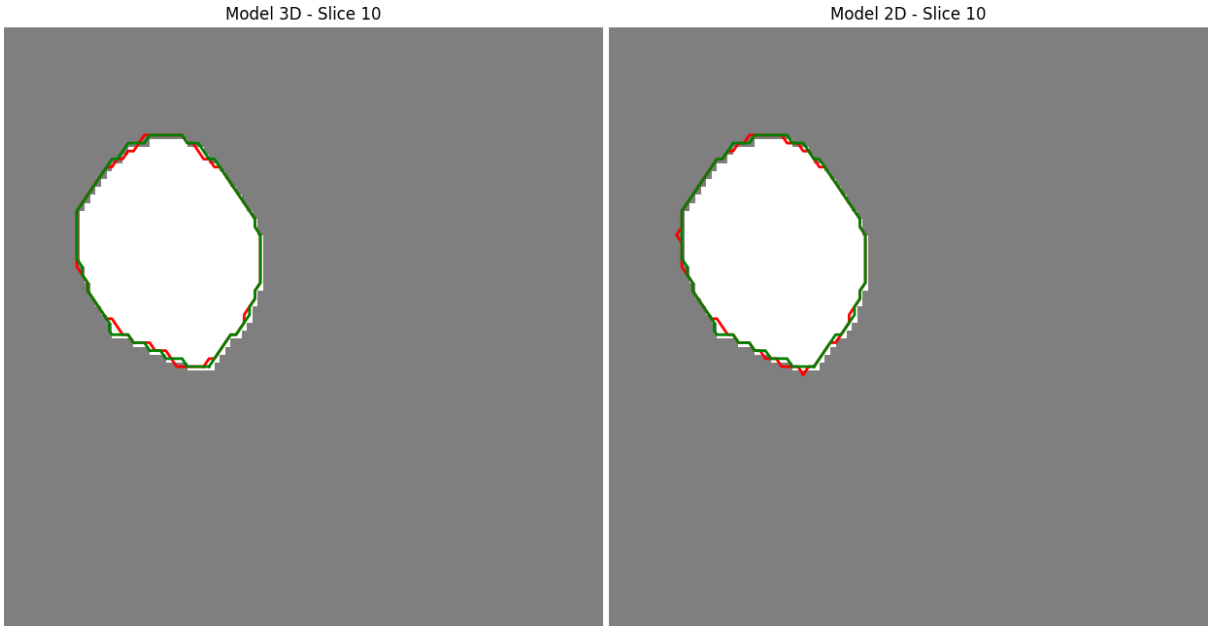

Figure 9: Contour overlay visualization of the 2D and 3D model segmentations compared to the ground truth for "F7L_fall.nrrd". The green contour depicts the contour of the ground truth labels, while the red contour depicts the contour of the predicted labels.

6.2.2 Original Labels

The two models, "2D, small, ResEnc" and "3D, small," were also trained on the full training dataset using the original labels generated by the "fill" algorithm. The evaluation on the test set is presented in Table 6. Initially there were concerns that unrealistic segmentations in certain regions, such as the C-shaped geometry of the femoral neck, might negatively affect the model's ability to segment accurately, but the performance scores were slightly better compared to those achieved with the manually corrected labels. This could be attributed to the fact that the manual corrections introduced more inconsistencies, as previously discussed, whereas the "fill" algorithm produces more rigid, rule-based segmentations, which the model appears to approximate well. Furthermore, the model's ability to learn and replicate these patterns, even when they involve labeling some non-bone pixels as bone, indicates great potential for the models to achieve even higher performance with more consistent, while still realistic, ground truth labeling.
When testing the models trained on the uncorrected labels against the manually corrected labels on the test set, the performance was worse compared to the models trained with corrected labels, as expected and shown in Table 6. The higher sensitivity and lower specificity highlight the tendency of the "fill" algorithm to label more pixels as bone compared to the manual correction. An example of the most significant differences between the predicted labels can be seen in Figure 10, where the model trained on the original labeling segments more pixels as bone inside the C-shaped femoral neck. At the distal regions of the femora, although the differences are less pronounced, there are still noticeable mismatches, as shown in Figure 11.

Model	**DICE**	**Sensitivity**	**Specificity**	**ASD (mm)**
2D	0.9892	0.9876	0.9985	0.1308
3D	0.9923	0.9909	0.9990	0.0922
2D Corrected Test	0.9770	0.9904	0.9942	0.3064
3D Corrected Test	0.9785	0.9923	0.9944	0.2878

Table 6: Mean performance metrics for 2D and 3D models on the test set using the original labels in training.

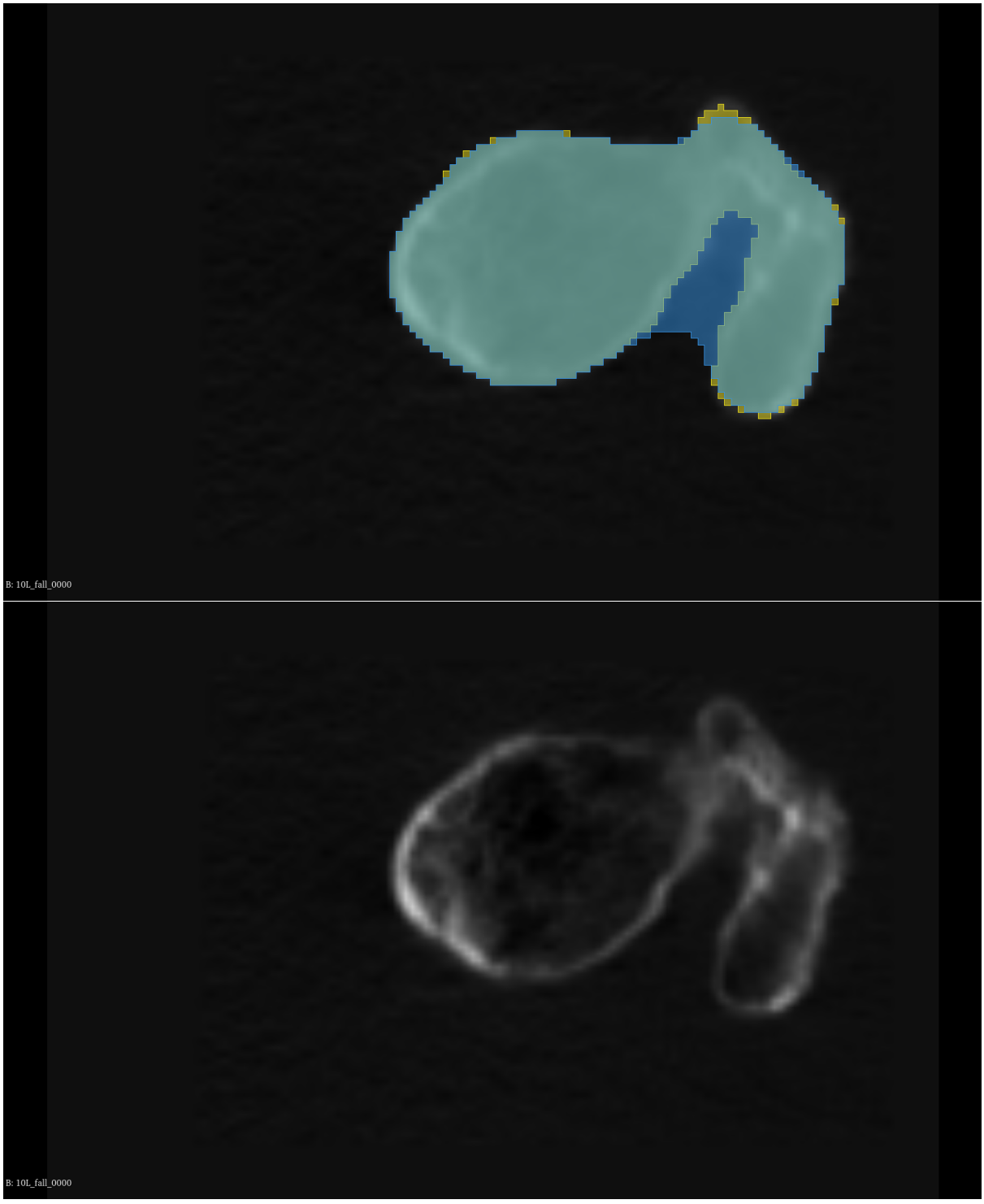

Figure 10: Top: 3D Slicer label overlay with the CT image as the background at the proximal region. Bottom: Raw CT image of the femur. Image name: "10_L_fall". Green pixels depict matching segmentation. Blue represents the labels predicted by the 3D model trained using the uncorrected labels as ground truth, while yellow represents the labels predicted by the 3D model trained using the corrected labels as ground truth.

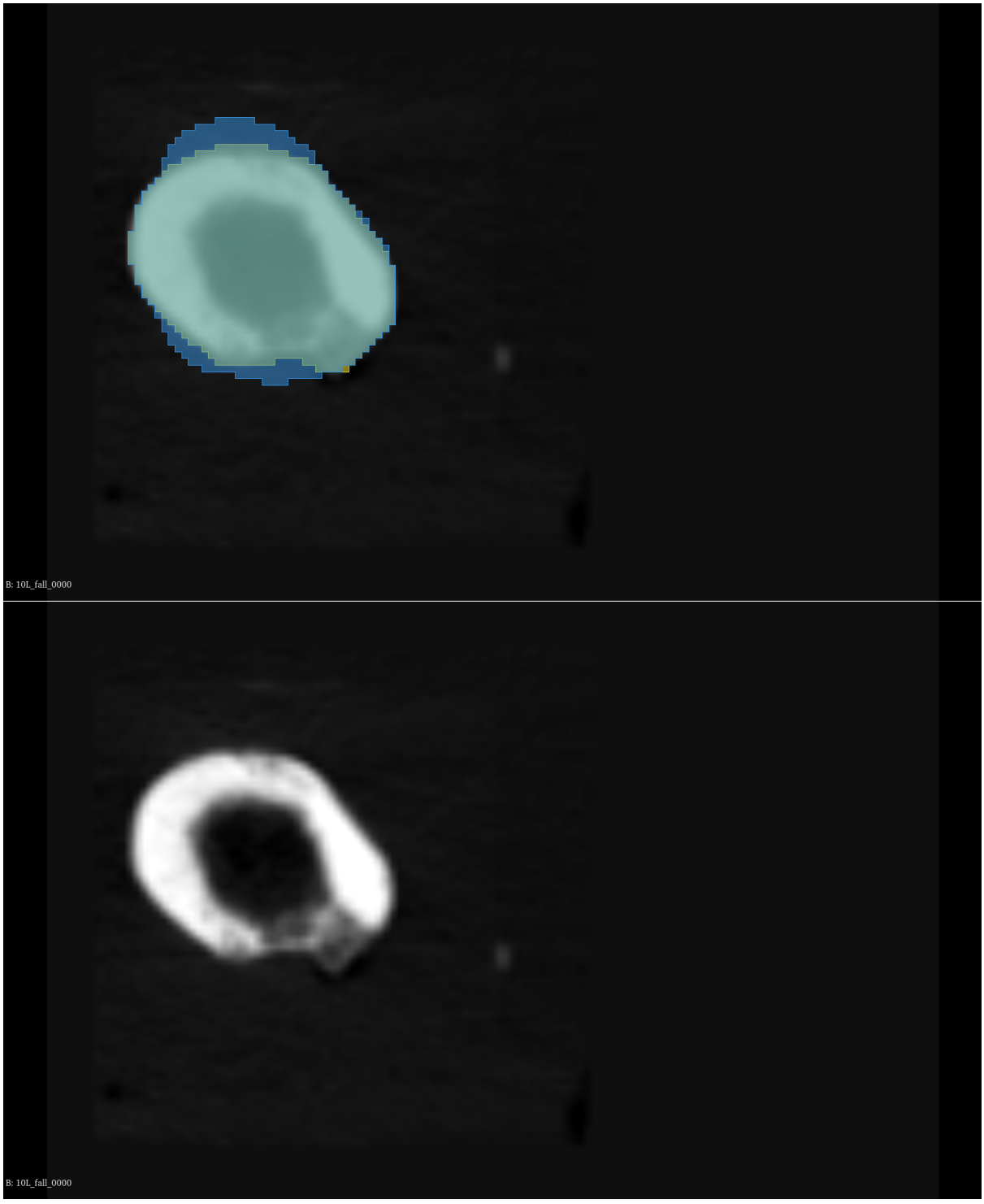

Figure 11: Top: 3D Slicer label overlay with the CT image as the background at the distal region. Bottom: Raw CT image of the femur. Image name: "10_L_fall". Green pixels depict matching segmentation. Blue represents the labels predicted by the 3D model trained using the uncorrected labels as ground truth, while yellow represents the labels predicted by the 3D model trained using the corrected labels as ground truth.

7 Conclusion

In conclusion, this project evaluated and optimized 2D and 3D nnU-Net models for femur segmentation in 3D CT images, while providing the code and respective documentation necessary for the reproducibility of the results [3]. Both models achieved high Dice scores, with the 3D model outperforming the 2D model across all performance metrics, including Sensitivity, Specificity, and Average Surface Distance (ASD). The parameter search showed no improvements from adjusting batch size and using data augmentation techniques, suggesting that the default nnU-Net configuration already approached near-optimal performance for this specific segmentation task. The Dice score of the best model was competitive with the baselines [2] [1], and the ASD surpassed them, despite using a smaller dataset. Additionally, the best model (3D) also offered the fastest inference times, least amount of used VRAM and the shortest training duration of all configurations.
The results also highlighted the challenge of inconsistent ground truth labeling, as the models performed slightly better with the rule-based labels generated by the "fill" algorithm when tested on the original labels. However, these models performed worse on the manually corrected test set, as expected. This indicates that, while manual corrections introduce inconsistencies, the model can adapt well to more consistent, rule-based segmentations. Hence, it can be concluded that further improvements are likely with more consistent, realistic ground truth labels and an expanded dataset.

References

[1] Zohar Yosibash et al. "Femurs Segmentation by Machine Learning from CT Scans Combined with Autonomous Finite Elements in Orthopedic and Endocrinology Applications". In: *Computers & Mathematics with Applications* 152 (Dec. 2023), pp. 16–27. DOI: 10.1016/j.camwa.2023.09.044. URL: https://doi.org/10.1016/j.camwa.2023.09.044.

[2] Yu Deng et al. "A Deep Learning-Based Approach to Automatic Proximal Femur Segmentation in Quantitative CT Images". In: *Medical & Biological Engineering & Computing* 60.5 (May 2022), pp. 1417–1429. DOI: 10.1007/s11517-022-02529-9. URL: https://doi.org/10.1007/s11517-022-02529-9.

[3] Jeremias Lang. *3D CT Proximal Femora Segmentation Using Optimized nnU-Net*. https://github.com/jereOG/femur_nnUNet. 2024.

[4] Dieter H. Pahr and Philippe K. Zysset. "From High-Resolution CT Data to Finite Element Models: Development of an Integrated Modular Framework". In: *Computer Methods in Biomechanics and Biomedical Engineering* 12.1 (Feb. 2009), pp. 45–57. DOI: 10.1080/10255840802144105. URL: https://doi.org/10.1080/10255840802144105.

[5] Jordi Minnema et al. "CT Image Segmentation of Bone for Medical Additive Manufacturing Using a Convolutional Neural Network". In: *Computers in Biology and Medicine* 103 (Dec. 2018), pp. 130–139. DOI: 10.1016/j.compbiomed.2018.10.012. URL: https://doi.org/10.1016/j.compbiomed.2018.10.012.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv* (2015). DOI: 10.48550/ARXIV.1505.04597. URL: https://doi.org/10.48550/ARXIV.1505.04597.

[7] Jun Ma et al. "Segment Anything in Medical Images". In: *Nature Communications* 15.1 (Jan. 2024). ISSN: 2041-1723. DOI: 10.1038/s41467-024-44824-z. URL: http://dx.doi.org/10.1038/s41467-024-44824-z.

[8] Ali Hatamizadeh et al. *Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images*. 2022. arXiv: 2201.01266 [eess.IV]. URL: https://arxiv.org/abs/2201.01266.

[9] Fabian Isensee et al. "nnU-Net: Self-Adapting Framework for U-Net-Based Medical Image Segmentation". In: *arXiv* (2018). DOI: 10.48550/ARXIV.1809.10486. URL: https://doi.org/10.48550/ARXIV.1809.10486.

[10] Jieneng Chen et al. *3D TransUNet: Advancing Medical Image Segmentation through Vision Transformers*. 2023. arXiv: 2310.07781 [cs.CV]. URL: https://arxiv.org/abs/2310.07781.

[11] Fabian Isensee et al. "nnU-Net Revisited: A Call for Rigorous Validation in 3D Medical Image Segmentation". In: *arXiv* (July 2024). URL: http://arxiv.org/abs/2404.09556.

[12] Fabian Isensee et al. *Extending nnU-Net is All You Need*. 2022. arXiv: 2208.10791. URL: https://doi.org/10.48550/arXiv.2208.10791.

[13] Yichi Zhang et al. "Bridging 2D and 3D Segmentation Networks for Computation Efficient Volumetric Medical Image Segmentation: An Empirical Study of 2.5D Solutions". In: *arXiv* (2020). DOI: 10.48550/ARXIV.2010.06163. URL: https://doi.org/10.48550/ARXIV.2010.06163.

[14] Benjamin Hohlmann, Peter Brößner, and Klaus Radermacher. "CNN Based 2D vs. 3D Segmentation of Bone in Ultrasound Images". In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. Accessed: August 22, 2024. 2024, pp. 116–110. DOI: 10.29007/qh4x. URL: https://doi.org/10.29007/qh4x.

[15] E. Dall'Ara et al. "A Nonlinear QCT-Based Finite Element Model Validation Study for the Human Femur Tested in Two Configurations in Vitro". In: *Bone* 52.1 (Jan. 2013), pp. 27–38. DOI: 10.1016/j.bone.2012.09.006. URL: https://doi.org/10.1016/j.bone.2012.09.006.

[16] Fabian Isensee et al. "nnU-Net: A Self-Configuring Method for Deep Learning-Based Biomedical Image Segmentation". In: *Nature Methods* 18.2 (Feb. 2021), pp. 203–211. DOI: 10.1038/s41592-020-01008-z. URL: https://doi.org/10.1038/s41592-020-01008-z.

[17] Fabian Isensee et al. *nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation*. https://github.com/MIC-DKFZ/nnUNet. Accessed: August 22, 2024. 2020.

[18] Ron Kikinis, Steve D. Pieper, and Kirby G. Vosburgh. "3D Slicer: A Platform for Subject-Specific Image Analysis, Visualization, and Clinical Support". In: *Intraoperative Imaging and Image-Guided Therapy*. Ed. by Ferenc A. Jolesz. New York, NY: Springer New York, 2014, pp. 277–289. DOI: 10.1007/978-1-4614-7657-3_19. URL: https://doi.org/10.1007/978-1-4614-7657-3_19.