An Automated Deep Learning based Approach for Hepatic Vessels Segmentation

Author:

Abdullah Afify

Abstract

This documentation explains how a classification task is implemented to segment hepatic vessels and tumors in medical images. Advanced deep learning methods, including the nnUNet framework and a traditional Convolutional Neural Network (CNN), to automate the identification of these important structures in medical images CT scans are used. Medical image segmentation is very important in healthcare, especially for analyzing liver blood vessels with CT scans. Accurate mapping is essential for liver surgeries and transplants. For liver surgeries, precise mapping helps plan and perform the operation, especially when tumors are involved. For living donor liver transplants, accurate mapping ensures the transplanted liver works well and keeps the donor safe. Radiologists find it challenging to analyze the complex structure of liver blood vessels, which is crucial for surgical planning, and the low contrast in CT scans makes this task even harder. However, automated image analysis with computer vision can make the process more efficient and accurate, improving diagnosis and surgical planning. The report describes two main approaches for the segmentation task. First, it uses the nnUNet, a deep learning method designed for biomedical image segmentation. nnUNet offers a flexible and efficient solution for accurately segmenting liver blood vessels and tumors. The report details the configuration, training process, and performance of nnUNet. Second, it explores two traditional CNN-based methods. These methods involve designing and training a custom CNN architecture for the segmentation task, providing more manual control over the network and its settings. This approach helps understand the basic principles of image segmentation and allows for fine-tuning the network. **It helped to get a Dice coefficient of about 86%.** Overall, this documentation serves as a guide for using advanced deep learning techniques to segment liver blood vessels and tumors in medical images, providing insights into both nnUNet and CNN-based methods.

Table of Content

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
CNN	Convolutional Neural Networks
nnUNet	A self-configuring method for deep learning-based biomedical image segmentation
CT	Computed Tomography
LDLT	Living Donor Liver Transplantations
SE	Squeeze and Excitation
AG	Attention Gates
ReLU	Rectified Linear Unit
FAIR	Facebook's AI Research lab
U-Net	CNN architecture for biomedical image segmentation image
V-Net	CNN architecture for volumetric medical image
SOTA	State-of-the-Art
GCN	Graph Convolutional Network
RF	Random Forrest
SVM	Supported Vector Machine
ASPP	Atrous Spatial Pyramid Pooling
CRFs	conditional random fields
SGD	stochastic gradient descent

Chapter 1

Introduction

1.1 Background

Medical image segmentation is very important in healthcare. It helps doctors analyze liver blood vessels using CT scans. This process allows doctors to see the liver's blood vessels clearly, making it easier to understand them. By doing this, doctors can make better decisions for treatments and surgeries. It is crucial for many procedures, like liver transplants from living donors and surgeries to remove parts of the liver. Accurate segmentation of liver blood vessels helps plan these surgeries and ensures the best outcomes for patients.

For example, in liver surgery, which is often needed for liver cancer, it's important to clearly identify and segment the liver's blood vessels. This helps in understanding the liver's structure, making surgical planning more precise. By carefully segmenting the liver's vessels and any abnormalities, doctors can evaluate things like the location and size of lesions and how they interact with the liver's blood vessels. This is crucial for deciding if surgery is possible.

A big worry is when liver tumors are closely connected to blood vessels. If tumors are tangled with these vessels, surgery can be very hard and might not be possible. So, it's really important to understand the blood vessel structure around liver tumors. This helps doctors make smart choices and give the best care to patients.

In liver transplants from living donors, it's very important to thoroughly examine the liver's blood vessels. This ensures that the transplanted liver works well and that the donor remains safe. Precise segmentation of the liver's blood vessels is necessary to assess if the vessels are compatible and to reduce the risks during the transplant process [1], [2].

Traditionally, segmenting medical images, especially liver blood vessels, has been done manually by expert radiologists. This method is time-consuming, requires a lot of effort, and can be biased. To address these issues, researchers are trying to automate this process. However, automating the segmentation of liver tissues and veins is complex.

There are many challenges, such as differences in liver anatomy between patients, the need for precise segmentation of blood vessels, low contrast in liver artery images, and irregularities caused by contrast injections. Overcoming these challenges requires new methods and advanced computer techniques to ensure accurate and reliable segmentation, improving diagnosis and making clinical workflows in liver imaging more efficient [3].

1.2 Motivation

The complex blood vessels in the liver are very hard for doctors to detect and understand. It takes a lot of time and work to figure out their shape and how they work. But, to plan surgeries well, doctors need to fully understand how these blood vessels are put together. This is especially important when there are problems with the blood vessels or when there are abnormal parts or tumors. In these cases, it's crucial to find and study veins and arteries very carefully to make sure surgeries go well. Another problem is that medical pictures, like ones from CT scans, often don't show all the details very clearly. This makes it hard for even the best doctors to make accurate diagnoses.

But, there's some good news. Computers using AI can help a lot with this. They can analyze liver blood vessels automatically, which saves a lot of time and work. This could change how diagnoses are made, making them faster and more accurate for surgery planning.

1.3 Research Key Questions

The primary research questions of this study are:

- What are the standard image preprocessing techniques that are appropriate for the enhancement of selected dataset?
- What are the best hepatic vessels segmentation techniques?
- Which are the benchmark deep CNN models that could be employed to perform the intended automatic segmentation task?
- How can deep learning based models be optimized for performing precise and efficient vessel segmentation across the selected medical imaging modality?
- What are the current challenges and future directions for performing the task of automatic vessel segmentation using deep learning?

1.4 Research Objectives

The main research objectives of current study have been briefed below:

- Assessing and comparing benchmark medical image preprocessing techniques tailored to the selected dataset to determine the optimal approach.
- Analyzing and contrasting state-of-the-art deep segmentation architectures, including ***UNet*** and various versions of ***nnUNet***, for semantic segmentation.
- Developing an optimized deep learning-based approach capable of efficiently automating hepatic vessel segmentation.
- Concluding by identifying key challenges inherent in current segmentation methodologies and proposing future directions for enhancing hepatic vessel segmentation.

1.5 Report Structure

This section provides an overview of the document's structure. The document is divided into four main chapters, each describing different aspects of our project. They are organized as follows:

- **Chapter 2 (Medical and Technical Background):**
  In this chapter, existing research and studies related to the project are reviewed. It brings together important findings, points out areas where more research is needed, and helps explain the background and importance of the current study.
- **Chapter 3 (Methodology and Implementation):**
  This chapter describes the Functional requirement, methodology of each step of the techniques and its implementation and the challenges faced in the process. Describing in detail the algorithms and techniques used to implement the project.
- **Chapter 4 (Experimental Results and Analysis):**
  This chapter explores the different paths that came across during the project, explaining why I chose the routes I did. It fully talks about the outcomes and results.
- **Chapter 5 (Conclusion and Future Directions):**
  In this chapter, project's findings and outcomes are summarized. There’s also suggestions and ideas for future research or developments based on this work.

Chapter 2

Medical and Technical Background

2.1 Hepatic Vasculature

Hepatic blood vessels are very important for the liver. They bring blood to and from the liver. Finding and diagnosing problems in these blood vessels is crucial for treating liver diseases. There are many types of liver blood vessel problems, like malformations, clots, narrowing, and growths. If these problems aren't treated, they can cause serious illness or death [19].

Finding and diagnosing liver blood vessel problems is important for several reasons. First, it helps doctors intervene early to prevent the disease from getting worse and causing other problems like liver failure, high blood pressure in the liver, or lack of blood flow. Second, getting the right diagnosis helps doctors choose the best treatment, like medicines, small surgeries, or big operations. Also, knowing exactly what kind of blood vessel problem someone has helps predict what will happen and how to treat them for a long time [20].

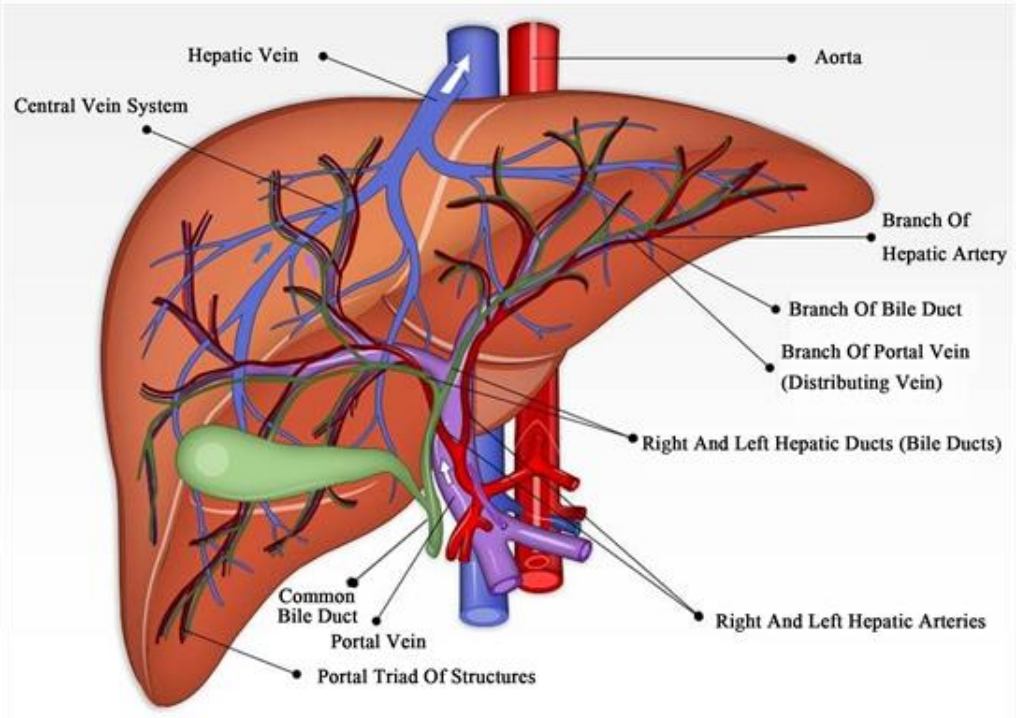


Figure 1 - The hepatic vasculature. [64]

Medical imaging methods like contrast-enhanced computed tomography (CT) and magnetic resonance imaging (MRI) are important for finding and diagnosing liver blood vessel problems. They give detailed pictures of the body's structure and show how blood flows through vessels. This helps doctors see and understand vascular issues like lesions with high accuracy. [21]

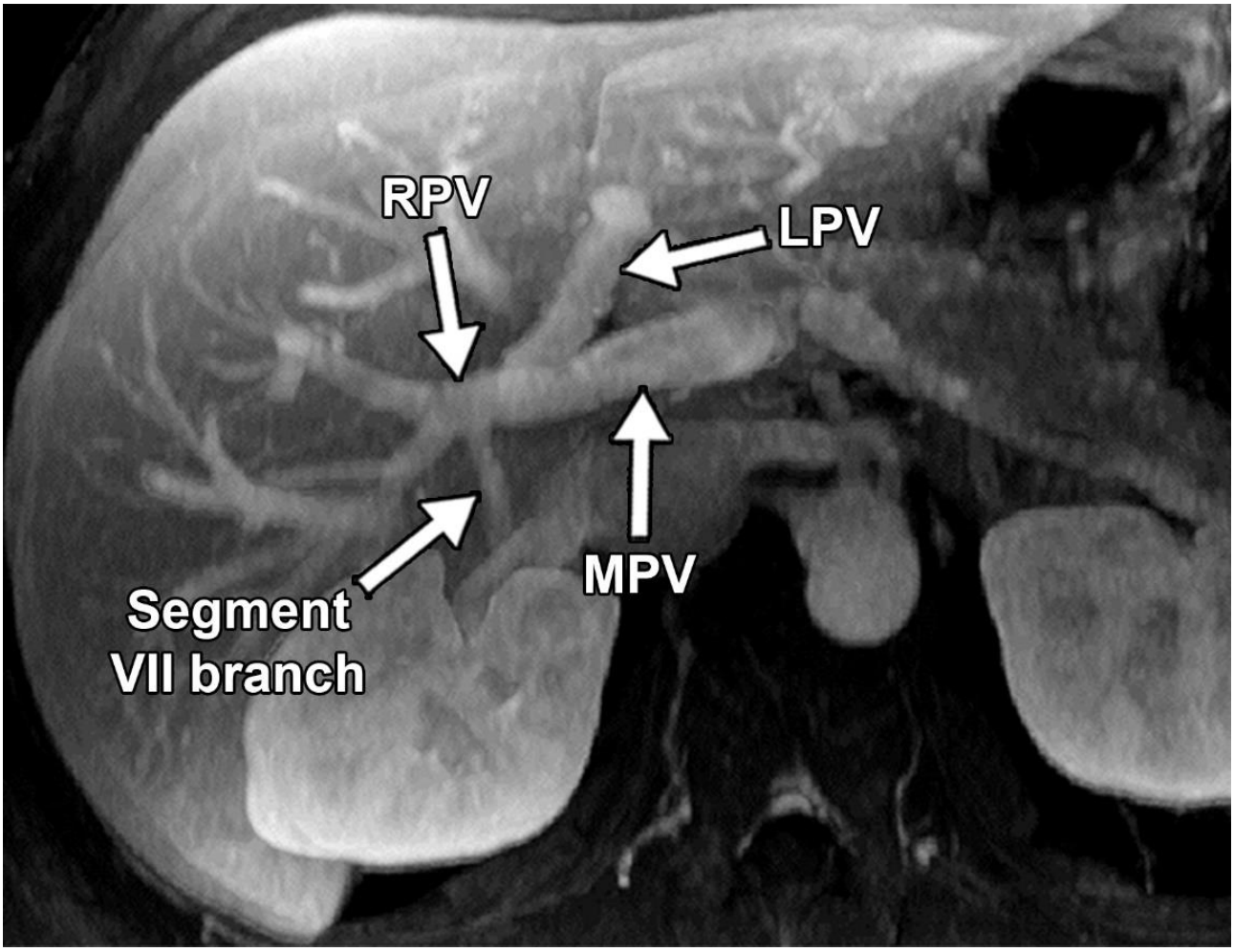


Figure 2 - Hepatic vasculature (CT imaging scans). [65]

2.2 State of the Art

Providing highly accurate and precise segmentation of liver blood vessels to doctors is a top concern for researchers who aim to improve surgical outcomes [4], [5]. Because of the many ways it can be used in clinics, a lot of work has been done to make it easier for doctors to see the liver's blood vessels. However, there's a big difference between general blood vessel and liver blood

vessel segmentation. Liver blood vessels are complex since they branch out a lot, and their size can be very small.

One way to segment blood vessels is the skeleton-based method. This method quickly finds and connects the middle parts of vessels to make a vessel tree. But, one big problem is that it needs a lot of help from the user [7]. *Shen and Soler* have looked at different **skeleton-based ways** to get blood vessel shapes [8]. Another way to segment blood vessels is the **region-growing-based** way. This way looks at how close or alike voxel parts are and puts the same ones together to make blood arteries [9]. There are different ways like this used in books to make it easier to do vessel segmentation. For example, one way called **context-voting** has been used with *CT pictures* to do vessel segmentation [10]. *Zeng and his friends* used a mix of **KMeans** and **3D region-growing** to find both thick and thin blood vessels [11].

These ways are very common, but they don't always give the best liver blood vessel shapes because they mostly look at how bright pictures are, and they don't think much about how blood vessels are put together [14]. To fix this, researchers have started to use deep learning. Deep learning can learn from very good pictures to find out more about blood vessels. Deep learning has helped a lot with doing blood vessel segmentation, and it's been especially good for liver blood vessels. Using things like **Convolutional Neural Networks (CNNs)** has made it much better at seeing things in medical pictures [15]. CNNs have been used in books to make very good pictures of bodies. For example, the U-Net way, with a different Dice loss, has been used to get liver blood vessel shapes without using a person. Also*, Agrawal and his friends* have made the **U-Net** way better by adding in new types of shapes and ways to look at things to find blood vessels [17]. *Kitrungrotsakul and his friends* made a deep **CNN way** to find liver blood vessels. They used deep CNNs that were made to look at parts and learn things about blood vessels from them [18].

2.3 Machine Learning

Machine learning (ML) is a field dedicated to developing algorithms capable of making predictions based on input data. By training on labeled datasets, ML algorithms create models that can identify patterns within new, unseen data. In the context of image analysis, such as in this case, training data may consist of labeled images, regions, or pixels. These patterns can range from low-level features, like pixel labels in segmentation tasks, to high-level concepts, such as the presence of a

hepatic vessel in medical images. Thus, image segmentation becomes the focus, with the training set comprising of images and their corresponding labels (vessels) [22].

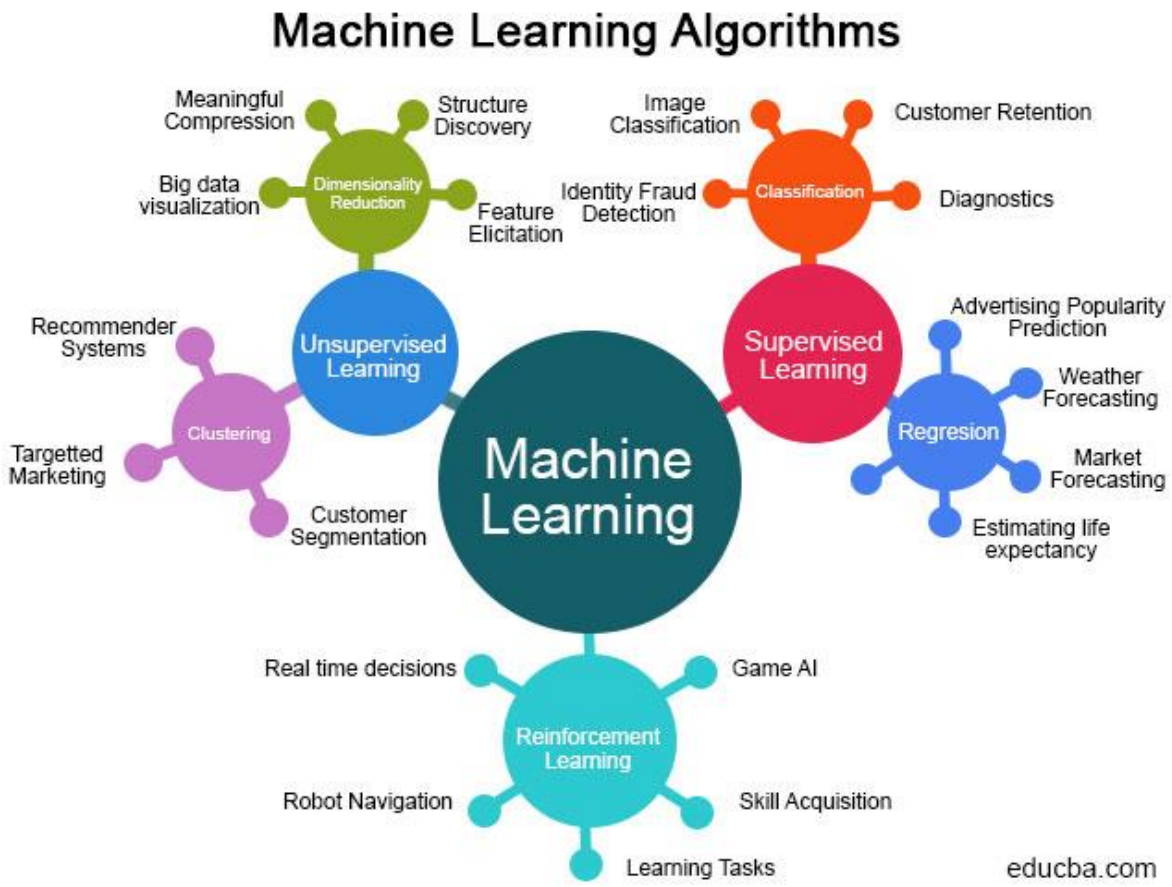


Figure 3 - Machine learning diagram. [66]

2.3.1 Neural Networks

The human brain's neural network is very complex and can do many tasks at the same time. In computer systems, we use a neural network (NN) to copy this complexity and act like a classifier, similar to the brain and its neurons. While biological neurons work differently, NNs use perceptrons as their basic units. The NN has different layers:

- Input layer for input features.
- Output layer for results.
- Middle layers of perceptron neurons connecting the input and output layers.

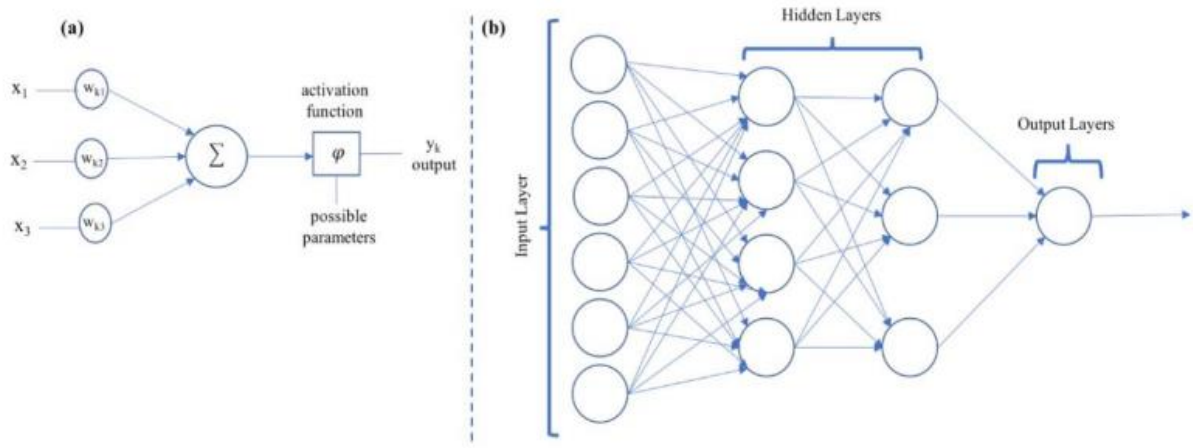


Figure 4 - (a) is the perceptron layer and (b) is the image of Multi-layer Neural Network. [67]

According to **the McCulloch-Pitts model**, a neuron, labeled as *'k'*, is fed with 'm' input parameters denoted by *'xj'*. Each input is paired with a weight parameter *'wkj'*. The neuron calculates the sum of the products of inputs and weights, which is then changed by an activation function *'φ'*, resulting in the output of the neuron *'yk'*, depicted in the accompanying diagram. This process is mathematically represented by the equation:

$$\boldsymbol{Yk = \phi(\sum jwkjxj)}$$

After training, a neural network can have the ability to approximate desired outputs by fine-tuning the weights of its neurons. Yet, finding the best weights for a multi-layer network is a complex analytical task. To tackle this issue, the **back-propagation algorithm** comes into play. This iterative procedure computes the gradient needed for adjusting weights. It consists of two primary stages: [27]

- Propagation.
- Weight update.

In the propagation phase, an input vector moves forward through the neural network, producing an output value. Then, the algorithm calculates the error term (cost). These error values sent backward through the network to assess the cost linked with hidden layer neurons afterwards.

In the weight update phase, the algorithm modifies neuron weights by determining the weight gradient and deducting a portion of it, called the learning rate, from the existing weights. This cycle continues with various inputs until the weights converge.

2.3.2 Deep Learning

Deep learning is a part of machine learning. It uses neural networks with three or more layers. These networks try to work like the human brain, but they are not as powerful. A neural network with one layer can make simple predictions, but adding more layers can make the predictions better and more accurate.

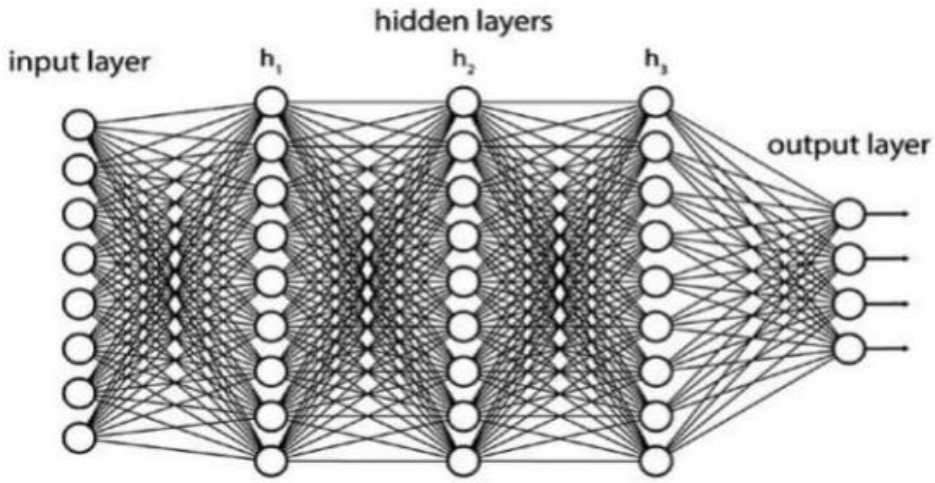


Figure 5 - Neural Network. [68]

Deep learning models work with multiple layers, which means they have more than two hidden layers in their neural networks. There are two types of deep learning models: **deep models** and **shallow models**. Deep models are better than shallow models.

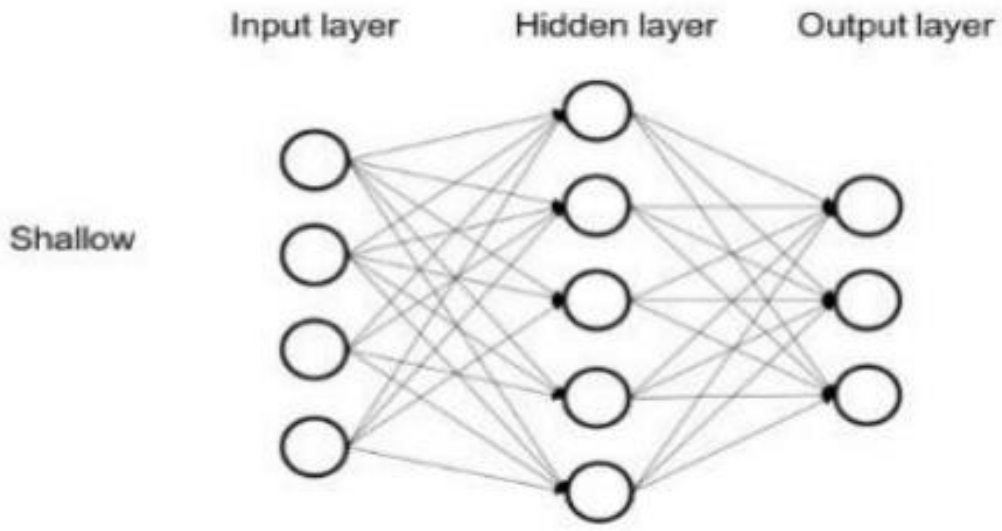


Figure 6- Shallow neural network. [69]

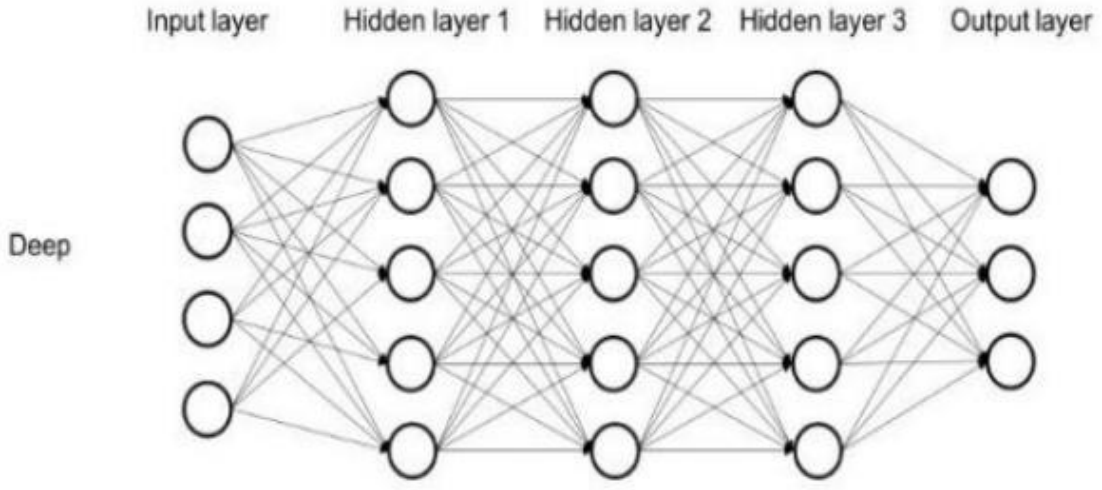


Figure 7 - Deep neural network. [70]

Deep neural networks are made up of many layers of connected nodes. Each layer improves predictions or classifications step by step, which is called **forward propagation**. The main layers

are the input layer, which takes in data, and the output layer, which gives the final predictions or classifications. [28]

Backpropagation, using methods like **gradient descent**, checks for errors in predictions and adjusts the weights and biases by moving backward through the layers to train the model. Forward propagation and backpropagation together help neural networks make predictions and fix mistakes, improving accuracy over time.

This basic structure of deep neural networks is part of deep learning, which includes many different algorithms for specific tasks or datasets. For instance, convolutional neural networks (CNNs) are used mostly for computer vision and image classification, as they are good at finding features and patterns in images. *In 2015*, a CNN even performed better than humans in an object recognition challenge for the first time. [29]

Benefits of Deep Learning over Machine Learning:

Deep learning is better at finding complex and global patterns in data than shallow learning methods. It also has other benefits, such as working well with large datasets, improving as more data is added, and automatically finding the best ways to represent the data.

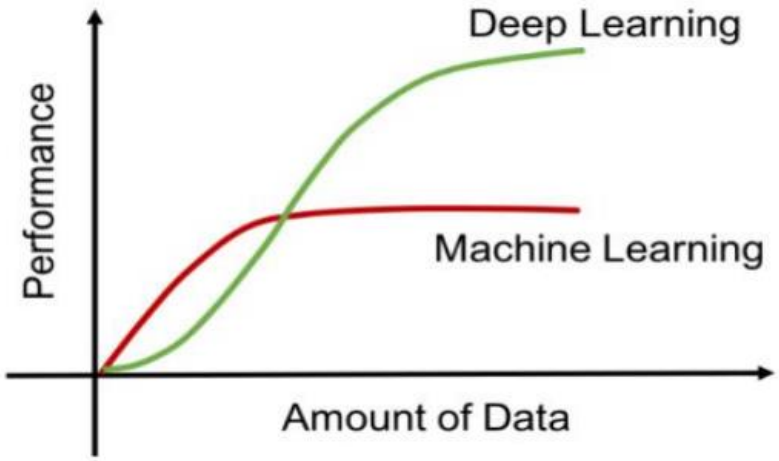


Figure 8 - Deep learning vs Machine learning. [71]

2.3.3 Artificial Neural Networks

The concept of artificial neural networks (ANNs) is inspired by the remarkable processing powers of the human brain and its complex neural networks [30]. ANNs are composed of interconnected units resembling artificial neurons, mimicking the operations of biological neurons in the human brain. These artificial neurons own the capability to transmit and receive signals from other neurons, thus processing information.

However, variations exist between ANNs and the human brain. ANNs are usually **static** and **symbolic**, whereas the human brain functions in an analog and dynamic manner. In ANNs, signals at connections are typically represented as real numbers, and the output of each neuron is determined by applying a non-linear function to the sum of its inputs.

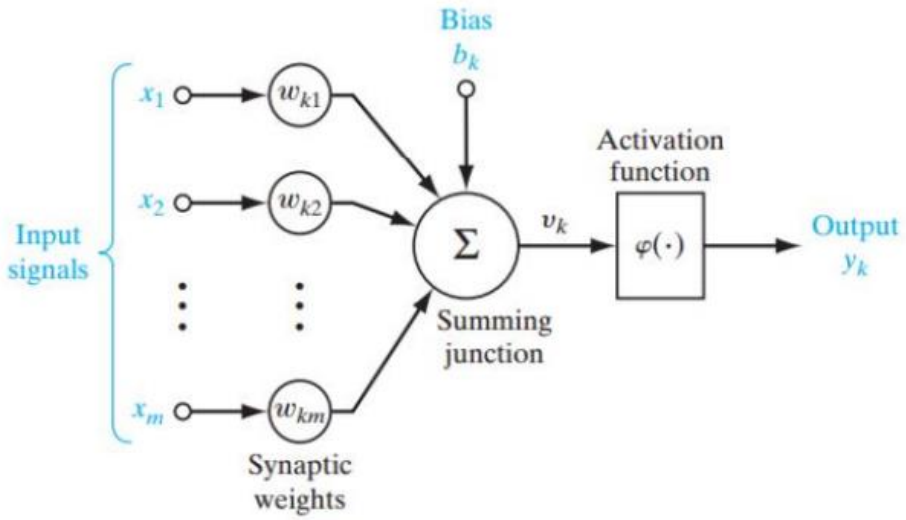


Figure 9 - Artificial neural network. [72]

2.3.4 Convolutional Neural Networks

In computer vision, convolutional neural networks (CNNs) are the top choice for neural network architecture. The reason is CNNs perform better than traditional neural networks (NNs) on computer vision tasks, especially with larger images. For example, a ***black-and-white*** image with 750x563 pixels has over 400,000 pixels, and a color image of the same size has over 1.2 million pixels. This leads to a lot of weights, which can cause over-fitting and lower performance in

traditional NNs. CNNs solve this problem by using shared weights and local connections, which reduces the number of parameters and the risk of over-fitting [31].

Moreover, CNNs need less image preprocessing compared to other image classification methods. They can learn important filters and features directly from raw data, so there's less need for manual feature engineering.

A typical CNN has input and output layers, along with several hidden layers. These hidden layers usually include convolutional layers, pooling layers, and fully connected layers.

- **Convolutional Layers:** These layers extract features from input data by employing convolution operations. Neurons within each layer mimic responses to visual stimuli, enabling the network to distinguish intricate patterns within images.
- **Pooling Layers:** These layers merges outputs of neuron clusters from preceding layers, effectively reducing the network's spatial dimensions and computational complexity while preserving essential features.
- **Rectified Linear Unit (ReLU) Layers:** These layers introduce non-linearity to the network through the application of the correct linear unit function, therefore enhancing the network's capability to highlight complex relationships within data.
- **Fully Connected Layers:** These layers establish connections between every neuron in one layer to every neuron in the subsequent layer, facilitating the learning of high-level features and classification.

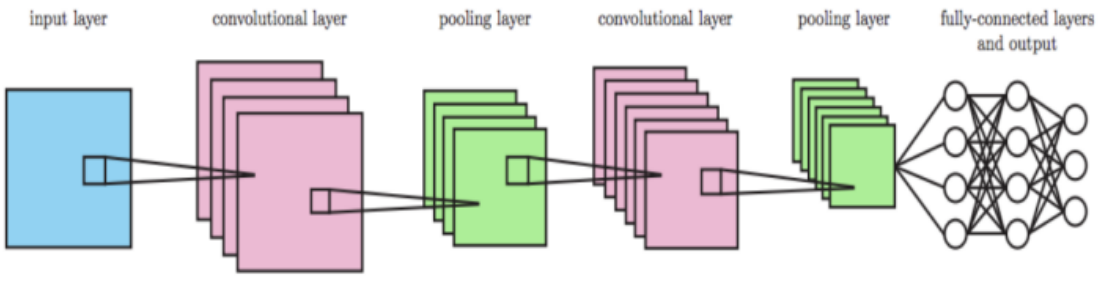


Figure 10 - An Example of a convolutional neural network. [73]

2.3.5 nnU-Net (Neural Network for U-Net)

Neural Networks for U-Net (nnU-Net) is a cutting-edge framework specifically designed for medical image segmentation tasks. Developed by *Isensee et al.*, nnU-Net represents a significant advancement in deep learning techniques applied to medical image analysis [32]. At its core, nnU-Net builds upon the foundational U-Net architecture, known for its effectiveness in biomedical image segmentation tasks. However, nnU-Net introduces several innovations and improvements to enhance performance and versatility [33].

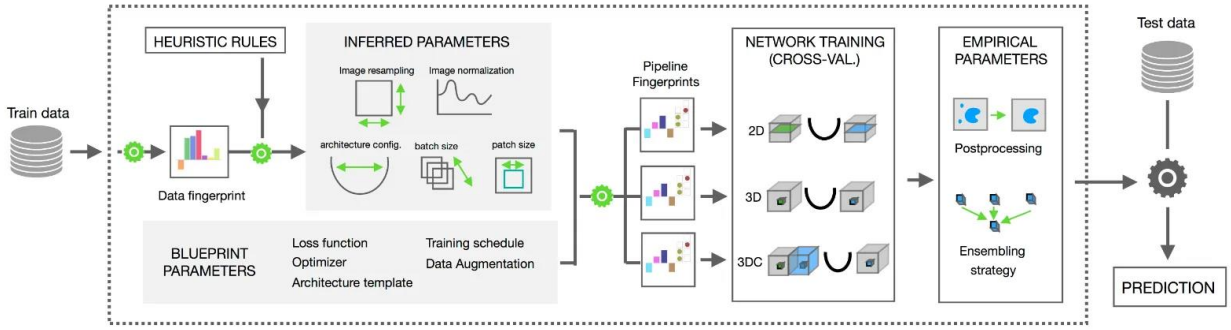


Figure 11 - nnUNet Complete Workflow. [74]

One of the key features of nnU-Net is its extensive pre-processing pipeline, which includes data normalization, intensity normalization, and data augmentation techniques designed for medical imaging data [34]. This pre-processing pipeline plays a crucial role in ensuring robust model training by standardizing input data and augmenting the training dataset to improve generalization.

Additionally, nnU-Net has a new network design that allows for automatic hyperparameter optimization, reducing the need for manual adjustments and making it easier for users of all skill levels [35]. This automatic optimization simplifies the model development process and improves the consistency of results across different datasets and tasks.

Moreover, nnU-Net uses the latest deep learning techniques, including advanced CNN architectures, regularization methods, and custom loss functions for medical image segmentation tasks [36]. By combining these techniques into one framework, nnU-Net performs better than traditional methods while staying computationally efficient.

In summary, nnU-Net is a major step forward in medical image segmentation, offering a strong and flexible framework for researchers and clinicians [37]. Its innovative features, like thorough pre-processing, automatic hyperparameter optimization, and advanced deep learning methods, make it highly valuable for various medical imaging applications.

2.4 Medical Image Segmentation

Medical image segmentation is a crucial process in healthcare that involves dividing medical images, such as MRI or CT scans, into meaningful parts. This helps doctors analyze and diagnose conditions more accurately. Segmentation is often done using advanced algorithms to highlight specific areas of interest, like organs or tumors, which aids in treatment planning and monitoring of patients' health.

2.4.1 Segmentation Algorithms

In recent years, there have been significant advancements in medical image segmentation. These developments are driven by the continuous improvement and exploration of advanced methods. Two main approaches have emerged**: semi-automated machine learning (ML) models** and **fully automated deep learning models**. These methods have revolutionized medical imaging by offering strong tools to extract important information from complex medical images.

2.4.1.1 Semi-Automated Segmentation Algorithms

Semi-automated segmentation algorithms are essential in medical image analysis. They combine computer methods with user help to segment images effectively [50]. These algorithms balance automation and human input, allowing users to correct errors and improve segmentation results for better accuracy. By involving users, these algorithms can adjust to different datasets, improving the quality of segmentation outcomes [51]. However, they depend heavily on user knowledge and input, which can introduce variation in the process. Also, the manual work needed can take time and slow down the analysis. Despite these challenges, semi-automated segmentation algorithms are vital in medical imaging, offering a good mix of efficiency and accuracy in segmentation tasks.

2.4.1.1.1 Random Forest (RF)

Random Forest is effective as it combines the knowledge of many decision trees to make strong predictions. Unlike a single decision tree that can be too specific and make mistakes, Random

Forest uses insights from different trees to make better guesses [52,53]. Each tree in the group is made separately, using a random part of the data and features. This randomness helps the trees look at the data in different ways, which stops them from being too similar and making the same mistakes.

Additionally, Random Forest uses ***bagging***, where each tree is trained on different parts of the data, to make the group stronger. The final guesses come from all the trees, with most guesses counted for picking classes and all guesses added up for numbers. This way of working, with randomness and bagging, makes Random Forest a useful and strong way to solve many machine learning problems.

2.4.1.1.2 Support Vector Machines (SVMs)

Support Vector Machine (SVM) is a strong tool for supervised machine learning used in sorting and estimating tasks. SVM tries to find the best line to split data points into different groups, making a gap called the margin between the line and the nearest points, called support vectors. By making this margin as big as possible, SVM can work well with new data it hasn't seen before.

Also, SVM can be used with non-straight data using kernel functions. These change the input data into more complicated ways, where a straight line is possible. You can see the plan of the SVM in *Figure 17*.

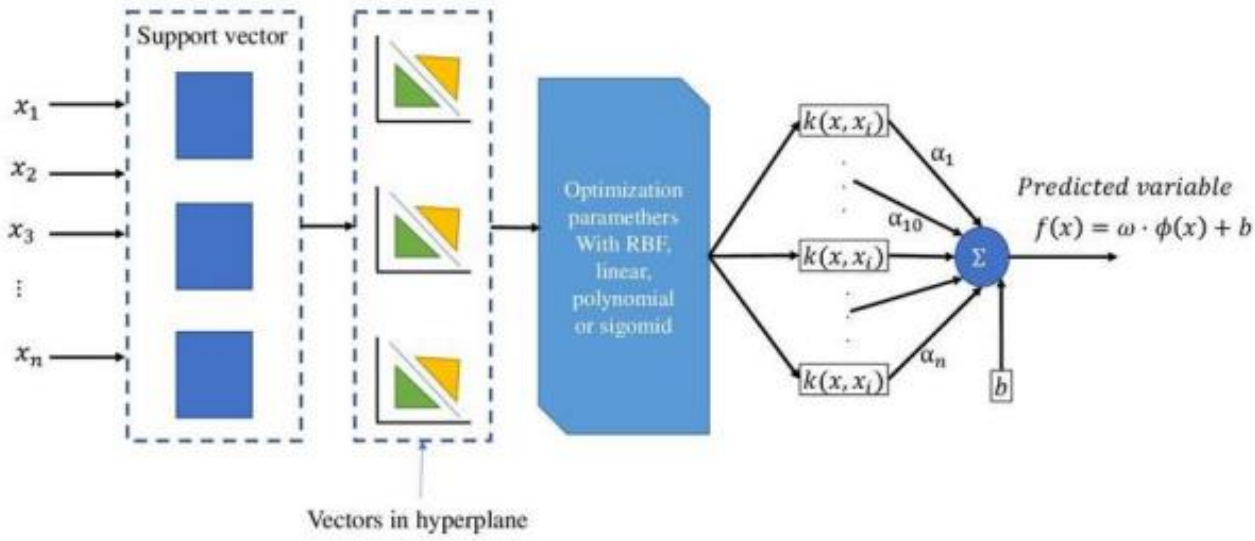


Figure 12 - SVM Architecture. [54]

In addition to its effectiveness in two-group classification tasks, SVM can handle multi-group classification using methods like **one-vs-one** and **one-vs-all** [55]. In the one-vs-one method, SVM creates many binary classifiers, each one designed to separate pairs of groups. When the system decides, the team's label is based on these binary categories' decisions.

2.4.1.1.3 Active Contour Tours

Active Contour Models, also known as snakes, are widely used in image processing and computer vision, especially for tasks like spotting objects and splitting images [56]. These models start a curve or outline near an object edge and change it bit by bit to show the object's shape. This shift is managed by reducing an energy form, which has inside power for soft curves and outside power from picture parts that make the curve go to the object edges. As the curve is updated over and over, active contour models smoothly catch object edges, even if there is noise or things in the way. Also, they can include user help to make the split results better and can work with different picture traits and object shapes.

These models are great at dealing with complex object shapes and picture traits [57]. By using both inside and outside power, active contour models smoothly show object edges, even if things are not clear or cover each other. Also, the cycle of the updates makes the split results better and more accurate. And so, active contour models have been added to other ways, like deep learning, to make them work better and more ways in many fields and uses. Due to this, active contour models stay an important thing in picture work and computer ways that need the right way to show objects.

2.4.1.2 Fully Automated Segmentation Algorithms

Fully automated segmentation algorithms operate independently of user interaction, providing a seamless and hands-off approach to segmentation tasks. These algorithms learn from training data to identify patterns and features within images. The primary advantage of fully automated segmentation is its efficiency and reduction in processing time. By eliminating the need for user involvement, these algorithms consistently produce reproducible results, enhancing reliability in medical image analysis. Additionally, their automated nature enables scalability and adaptability to large datasets, making them particularly beneficial for tasks requiring high throughput or handling extensive data volumes. Unlike semi-automated segmentation techniques, fully

automated methods are generally faster and more efficient, especially in large-scale or high-throughput applications where manual intervention may be impractical or time-consuming.

2.4.1.2.1 U-Net

The U-Net architecture is highly regarded in biomedical image segmentation, particularly in medical image analysis. Its special U-shaped structure consists of an encoding (slimming) and decoding (spreading) part. It's designed to create precise segmentations by capturing local and global features.

In the slimming part, U-Net functions as a feature extractor, capturing important aspects of input images with a series of convolutional and pooling layers. This reduces spatial dimensions while preserving important features, enabling segmentations that contain both local and global information.

In the spreading part, U-Net focuses on recovering spatial details and generating the final segmentation map. It uses upsampling layers to enhance spatial resolution.

U-Net establishes connections between the slimming and spreading paths to preserve fine details. By minimizing information loss, this approach can produce accurate segmentations.

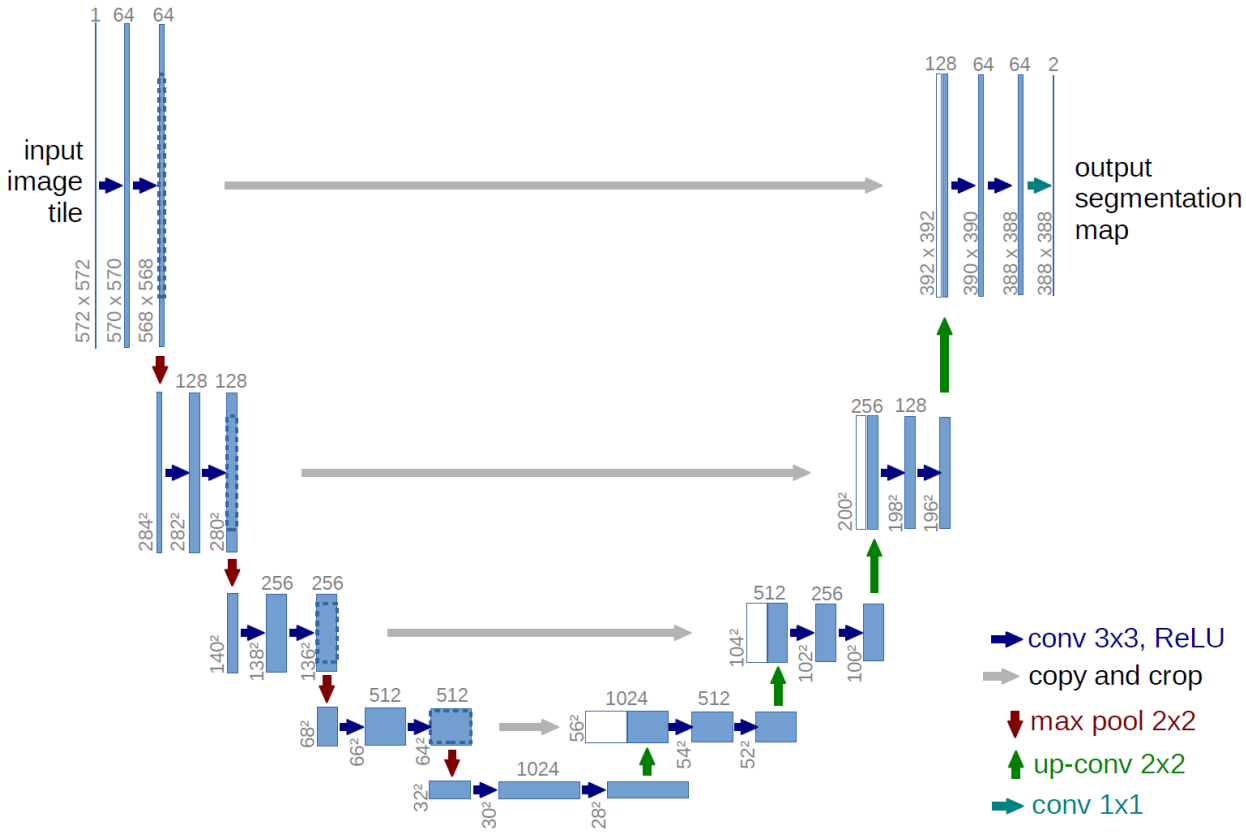


Figure 13 - U-net Architecture. [58]

2.4.1.2.2 DeepLabv3+

DeepLabv3+ is a powerful type of computer system that is made for recognizing the meaning of images. One of its most important new ideas is how it can see both small and big details at the same time. This makes it very good at knowing exactly what each part of a picture is.

In the picture shown in *Figure 19*.

DeepLabv3+ looks like a special kind of math machine that's built for recognizing the meaning of images. At its center, the system uses a main computer part, often made with math tools like **ResNet** or **MobileNet**, to pick out important parts from images. These parts are used as the bottom layer to make the system understand very tricky image things. [59]

One important part, **the Atrous Spatial Pyramid Pooling (ASPP) tool**, is used to get more than one level of image information at the same time. This makes it possible for the system to get very

close and very far away things right. The tools that change the picture in different ways are used in both the main computer part and ASPP tools to get more of the picture in one go. This is how the system can keep the picture just as clear while getting more things around it. The system then gets the picture back to the way it started and takes away things that don't go with the picture.

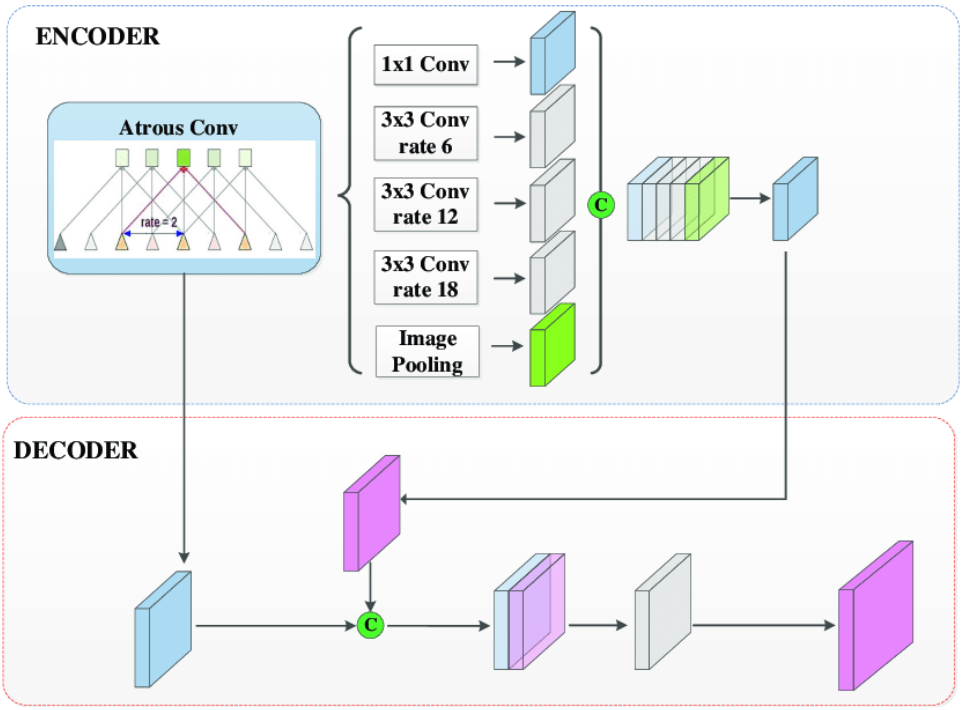


Figure 14 - DeepLabv3+ Architecture. [75]

DeepLabv3+ doesn't just rely on its smart design; it also uses advanced methods to make it work even better. Normally, the system is trained using strong methods like **stochastic gradient descent (SGD)** or **Adam**. These are helped by things like changing how fast it learns and keeping up the speed.

Also, D**eepLabv3+** gets a good start by learning from big sets of pictures like **ImageNet**. This helps the system learn lots of good things about pictures, which it can use to do well on lots of different jobs. After that, it learns more about specific jobs by training on sets of pictures made just for those jobs. This helps the system get even better at understanding the special parts of different kinds of pictures.

Overall, **DeepLabv3+** combines its smart design with good ways to make it work better. This makes it a top choice for understanding what's in pictures, and it does well in many jobs in computer vision and other areas, always doing a good job.

2.4.1.2.3 Graph Convolutional Network (GCN)

Graph Convolutional Networks (GCNs) are a new and promising way to split images, giving the ability to mix parts of a picture and details about how they connect. By setting up a map where each dot shows a pixel or part of one and the lines show how close they are, GCNs can easily send and mix information across the whole picture. This makes the results of breaking up a picture a lot better.

The way a graph convolutional network works for splitting up pictures usually goes through a few main steps, as shown in *Figure 20*:

1. **Making the Map:** First, a map is made from the picture, with each dot showing a pixel or part of one, and the lines showing how close they are to each other.
2. **Starting the Dots:** At this point, each dot gets started with what it needs to know from the picture, which is used to help later on.
3. **Sending Messages and Mixing:** By doing this over and over, the GCN puts together what it learns from the dots next to each other. This lets each dot make what it knows about the picture better, by getting a mix of information that helps it see how parts of the picture are close and connected.
4. **Making Predictions:** After a few times of sending messages and mixing what's there, the dots are used to guess what each part of the picture is. This last step can mean saying what kind of thing is in each part, or making a big plan of what each part of the picture is.

This way of working shows how good GCNs are at using maps to split pictures up with a lot of detail and without wasting time.

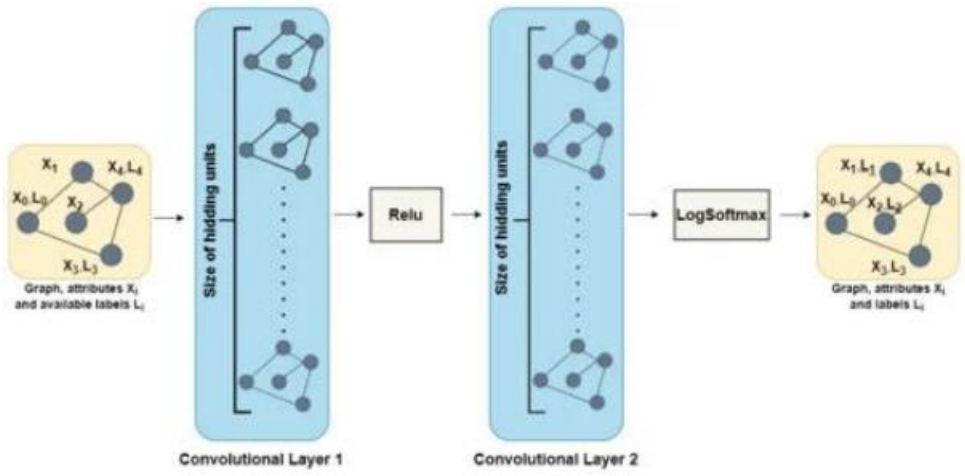


Figure 15 - Working of a Graph Convolutional Network. [61]

2.4.1.3 Recent Advances and Trends

This section will discuss the resent advanced trends in Medical Image Segmentation.

2.4.1.3.1 Multi-modal Segmentation Techniques

Multi-modal segmentation is an important strategy in medical image analysis, especially in fields like radiology and neuroimaging, where various types of imaging such as MRI, CT, PET, and ultrasound are used. This approach, as explained by references [62], combines information from different types of imaging to improve segmentation accuracy and reliability.

By combining data from different imaging perspectives, multi-modal segmentation provides several benefits:

1. **Improved Segmentation Accuracy:** Integrating information from different imaging types helps better understand and outline anatomical structures or pathological areas.
   The different types of information compensate for the limitations of individual imaging types, resulting in more precise segmentation.
2. **Increased Robustness:** Multi-modal segmentation improves resistance to noise, artifacts, and variations in medical imaging data.
   Issues or uncertainties in one type of imaging can be reduced by information from other types, reducing the impact of specific limitations on segmentation.

3. **Comprehensive Assessment:** By using the strengths of each imaging type, multi-modal segmentation provides a broader view of the anatomy or pathology being studied.
   This thorough assessment helps clinicians and researchers confirm findings across different imaging types, giving more confidence in segmentation results and helping in making informed decisions for medical diagnosis and treatment planning.

2.4.1.3.2 Transfer Learning and Domain Adaptation

Transfer learning and domain adaptation are important strategies in image segmentation, especially when dealing with datasets from different domains or imaging types.

Transfer learning involves using knowledge gained from training on a source domain with lots of labeled data to improve performance on a target domain where labeled data is scarce or missing. In image segmentation, this means fine-tuning pre-trained models, originally trained on large datasets like ImageNet, for the specific segmentation task. This allows the model to transfer learned representations to the new domain.

Domain adaptation focuses on adjusting the model to the target domain by addressing the differences between the source and target domains. This can include approaches like adversarial training, where the model is trained to reduce the differences between the distributions of the source and target domains. Domain adaptation might also use domain-specific data augmentation methods, creating artificial samples to mimic the target domain's attributes.

Additionally, unsupervised or semi-supervised domain adaptation uses unlabeled data from the target domain to improve the model's representations and bridge the domain gap. These methods often focus on learning domain-invariant features, helping the model generalize well to unseen data from the target domain.

These techniques enhance the adaptability of segmentation models across different domains and improve their performance when labeled data is limited or unavailable, making them very useful in real-world applications.

2.4.2 Hepatic Vessels Segmentation

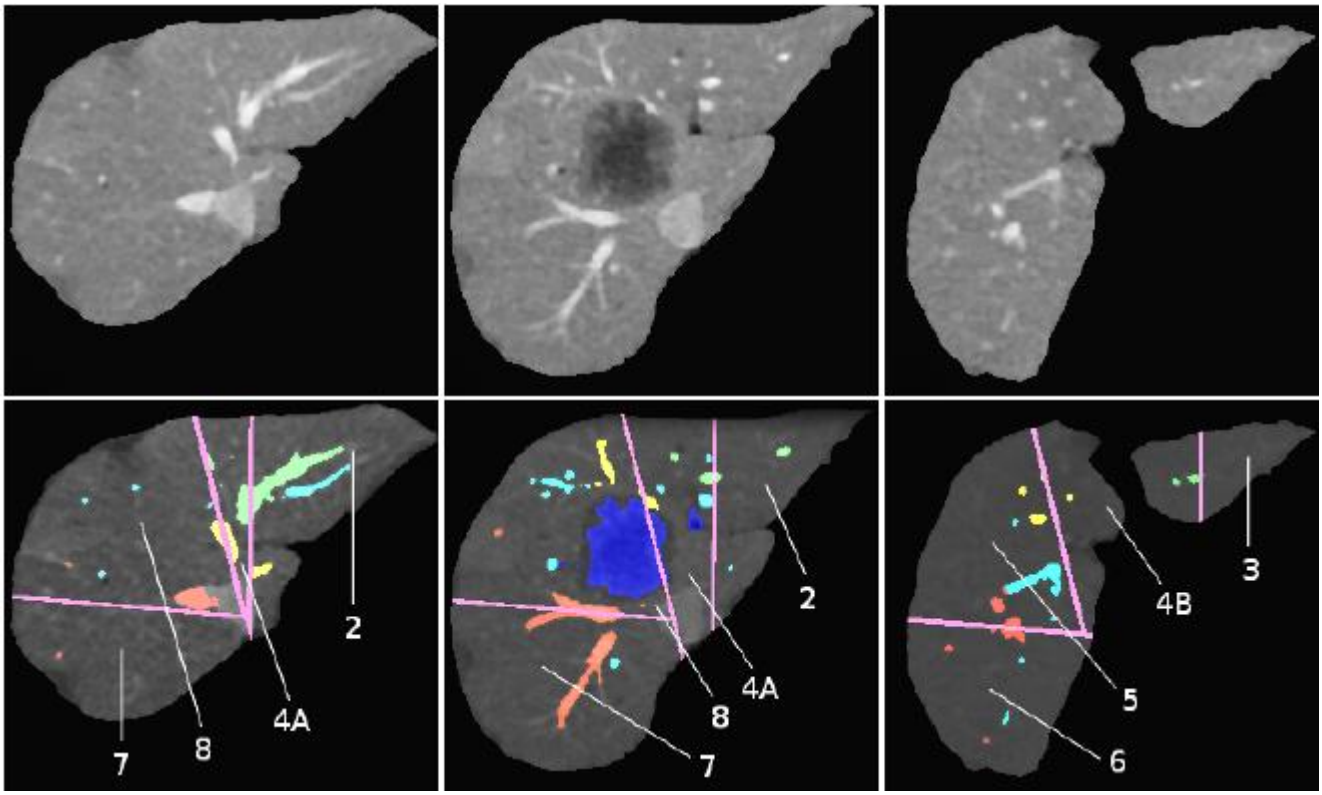


Figure 16 - Hepatic Vessels Segmentation (Veins & Tumors). [76]

Hepatic vessel segmentation is an important task in medical imaging that involves identifying blood vessels in the liver from imaging data like CT scans. Accurate segmentation is crucial for clinical applications such as surgical planning, diagnosing liver diseases, and monitoring treatments. However, this task is challenging due to the complex structures of the vessels, their different shapes, and their low contrast with surrounding tissues.

To overcome these challenges, researchers have developed advanced segmentation algorithms, often using deep learning techniques like CNNs. These models use large datasets to learn complex patterns and features of hepatic vessels, allowing for automated and precise segmentation. Improvements in image preprocessing, feature extraction, and post-processing techniques also help increase segmentation accuracy and reliability.

Effective segmentation of hepatic vessels supports clinical decision-making and helps advance liver-related medical research and treatment methods.

2.4.2.1 Challenges and Limitations in Current Segmentation Approaches

Current methods for segmenting hepatic vessels face significant challenges, including complex vessel structures and low tissue contrast. These difficulties often result in less accurate segmentation. Improving these methods is important for better accuracy and reliability, which enhances clinical decision-making and patient care.

2.4.2.2 Deep Learning in Medical Image Segmentation

Deep learning techniques, especially CNNs, have become powerful tools for medical image segmentation. They offer promising solutions to the problems in hepatic vessel segmentation. By automatically learning important features from data, CNNs can understand the complex structures of hepatic vessels and improve segmentation accuracy. [44].

2.4.2.3 Advanced Segmentation With nnU-net

Building on the success of deep learning, nnU-net is a significant advancement in medical image segmentation, including for hepatic vessels. Its innovative design, with instance normalization and dense skip connections, allows for more accurate segmentation of complex vessel structures. Using nnU-net can lead to better segmentation results and improved clinical outcomes for patients.

2.4.2.3.1 nnU-Net's Contribution to Addressing Segmentation Challenges

nnU-net's unique features, such as instance normalization and dense skip connections, directly address the challenges in hepatic vessel segmentation. Its ability to capture complex structures and adapt to different imaging conditions makes it a valuable tool for improving segmentation accuracy and reliability. Using nnU-net can provide more reliable clinical insights and better patient outcomes. [49]

2.5 Dataset Details (Dataset101_HepVes)

Hepatic vessel datasets are collections of medical imaging data related to the hepatic (liver) vessels. This dataset typically includes various types of imaging modalities such as CT scans.

The purpose of this dataset is to aid in the development and evaluation of algorithms and techniques for tasks such as vessel segmentation, detection, and classification. The dataset after organization is structured as follows:

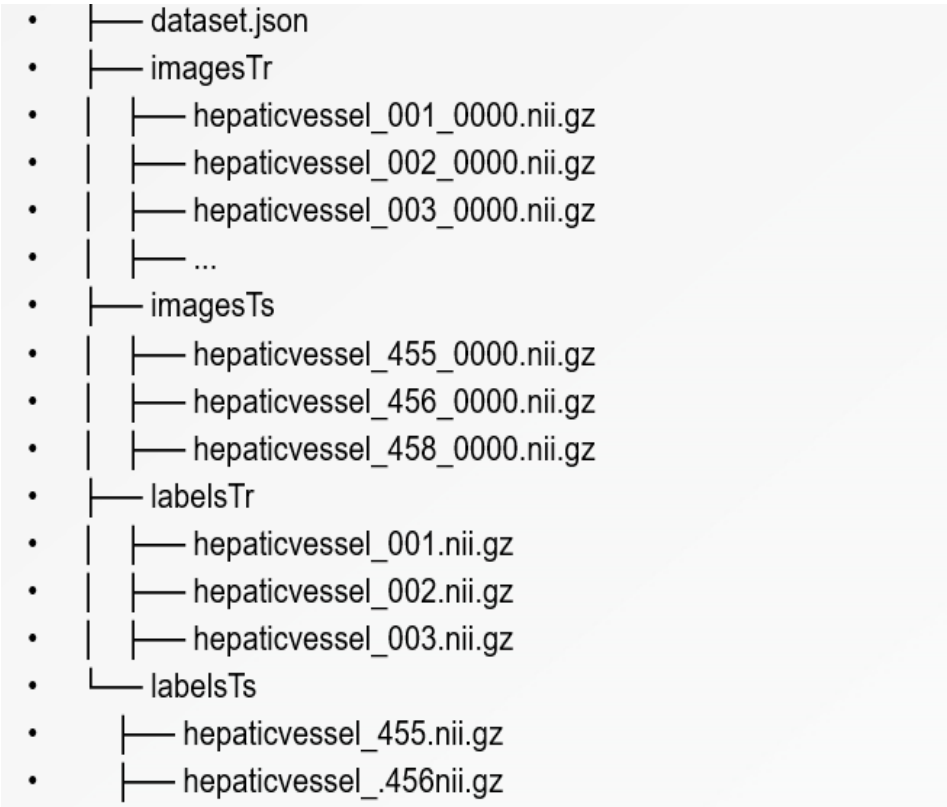


Figure 17- Dataset folder structure (Dataset101_HepVes)

The data is in the form of **'nii.gz'** format, which will be suitable for the technique structure discussed in detail in the upcoming chapters.

Here are some examples visualizations of the data set in the next three figures:

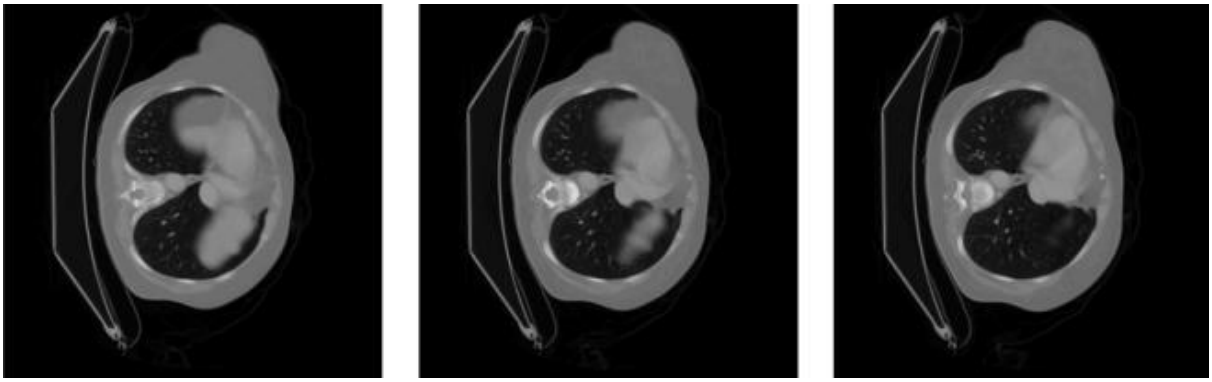


Figure 18 - training images.

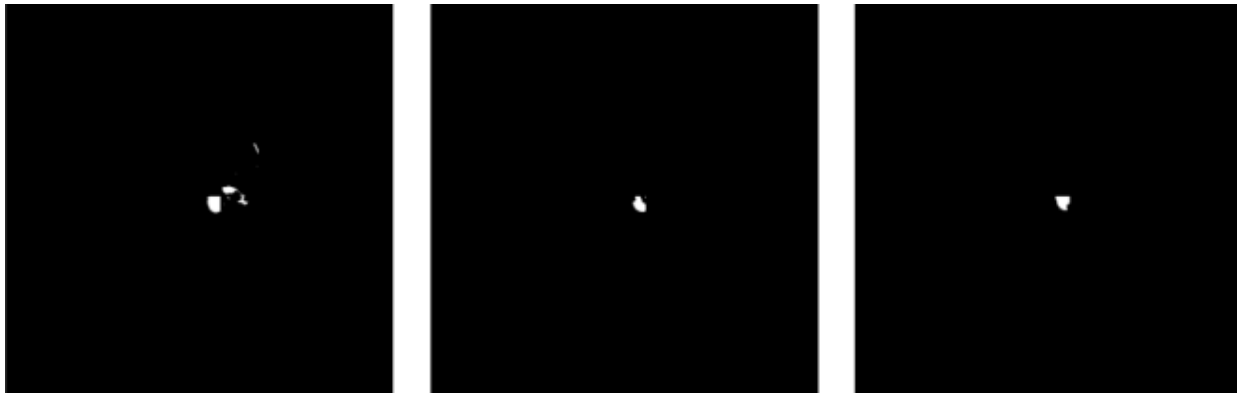


Figure 19 - training labels.

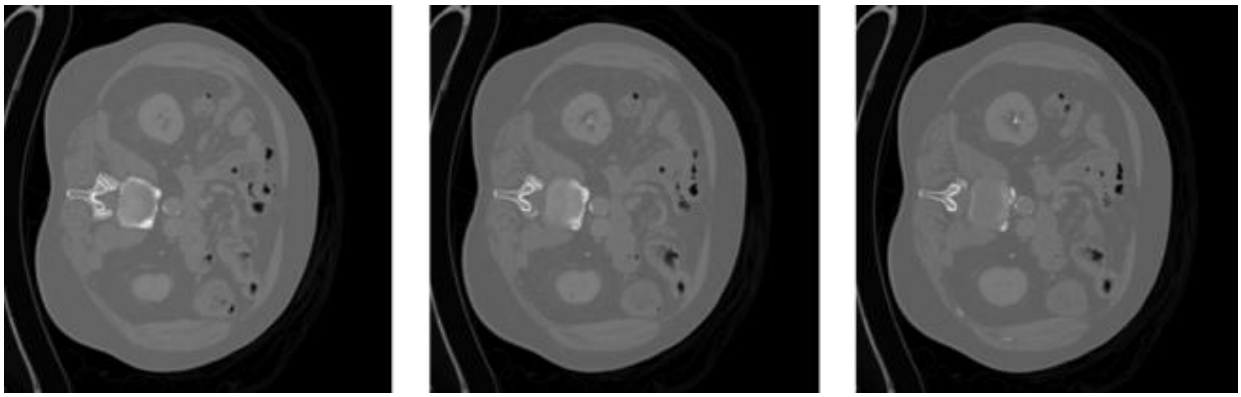


Figure 20 - test images.

Figure 21 - test labels.

Chapter 3

Methodology and Implementation

3.1 Identifying Optimal Strategies: Exploring Three Techniques

When preparing for this task, we faced a crucial decision in the planning phase: choosing the right model or tool. This choice was especially important since we had a limited dataset of only 303 images, with 260 used for training. With many methods available, we had to carefully consider factors like computational efficiency, adaptability to our dataset size, and proven effectiveness in similar tasks. To make an informed decision, we thoroughly reviewed existing literature and research papers. After extensive exploration, we chose the nnU-Net methodology.

nnU-Net stood out due to its unique features and capabilities. It is a high-demand computation self-configured technique, suitable for our task. Its ability to adapt to different dataset sizes and handle complex image segmentation tasks fit our project needs. However, we didn't choose nnU-Net in isolation. We compared it with the traditional CNN U-net approach, which uses standard preprocessing and planning techniques. This comparison helped us assess nnU-Net's effectiveness and see its unique contributions and optimizations compared to conventional methods.

By contrasting nnU-Net with the traditional CNN U-net approach, we committed to rigorous experimentation and validation. This method, based on scientific principles, aims to highlight the differences and benefits of each approach, and consider combining them for the best results. Our comparative analysis shows the dynamic nature of research and innovation in machine learning and image processing. As we explore and discover, we aim to contribute to the advancement of knowledge and improve techniques in image segmentation and beyond. [41]

3.2 nnUNet Technique

nnUNet is a powerful tool for medical image segmentation, even with small datasets. It uses a deep learning architecture with a series of neural networks for strong segmentation performance. By using techniques like data augmentation, transfer learning, and ensembling, **nnUNet** effectively handles the challenges of limited data. Introduced by Isensee et al. in their paper "**nnUNet**: A self-configuring method for deep learning-based biomedical image segmentation,"

nnUNet helps researchers achieve competitive results with small datasets, making it valuable for medical image analysis where data is scarce. [42]

nnUNet offers different configurations for various needs in medical image segmentation. These include **2D** and **3D** networks and different architectures like **U-Net** and **V-Net**. Users can adapt **nnUNet** to their specific dataset and computational resources. Additionally, **nnUNet** allows flexibility in preprocessing, postprocessing, and training strategies, helping researchers optimize performance based on their unique needs. This versatility is highlighted by Isensee et al. in their paper, where they show the effectiveness of different configurations across various medical imaging datasets and tasks.

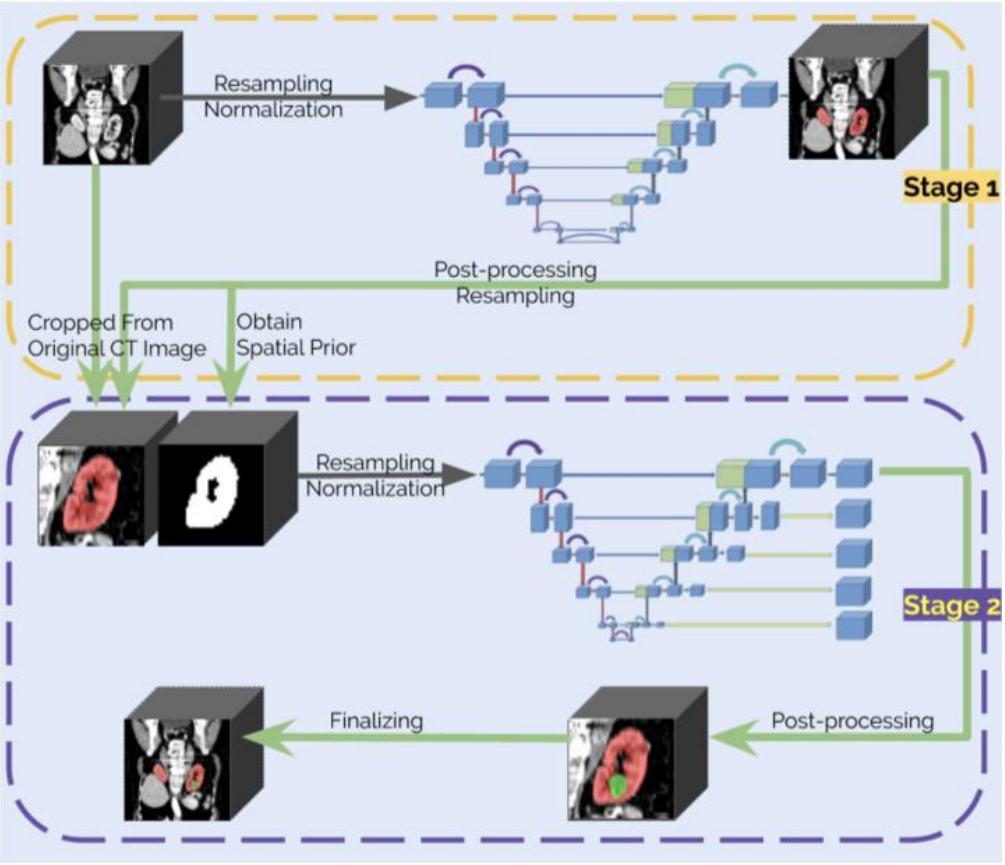


Figure 22 - nnUnet strategy stages. [77]

3.2.1 nnUNet Configuration and Setup

Setting up nnUNet for my medical image segmentation task involved several steps. First, dataset was prepared by organizing the images and labels into the required format and directory structure.

Next, a suitable configuration was selected for task, choosing from options like network architecture (e.g., U-Net), preprocessing techniques, and training strategies. Then, configuration parameters were customized to match my dataset and computational resources.

After configuring nnUNet, the model was trained with my dataset, monitoring performance metrics and adjusting parameters to optimize segmentation accuracy. Once trained, it was evaluated the model on a separate validation set to check its generalization capabilities. Finally, the trained model was deployed to segment new medical images, using nnUNet's efficient inference capabilities to produce accurate results.

Throughout the process, nnUNet's comprehensive documentation and support helped me execute smoothly, allowing me to focus on my specific biomedical imaging challenges.

3.2.1.1 Prepare the Environment

nnU-Net needs a lot of computing power and storage, so we decided to use *Google Colab Pro* and *Google Drive*. To make it faster, we set it up to use GPU acceleration with the **'CUDA'** setting in *PyTorch*, instead of just the CPU, which was slow during each epoch. Using the GPU made it much faster since *Google Colab Pro* has powerful GPUs like the A100 and V100. But this efficiency has a cost, as the A100 GPU uses more than 11 units per hour. We tried to enable hyper mode to use both CPU and GPU together, but nnU-Net doesn't support this. All these settings are in the ***'run-training.py'*** file in the main directory of the nnU-Net 'Run folder'.

```
parser.add_argument('-device', type=str, default='cuda', required=False,
```

Figure 23 - make Cuda the default option.

The settings in the provided code was adjusted to make the model run better. At first, the code sets global options like the number of processes, anisotropic data threshold, and convolution kernel sizes. These settings are stored in a default configuration dictionary. Two main functions, ***save_configuration()*** and ***load_configuration(),*** help manage these settings**.** ***save_configuration()*** saves the default settings to a specified JSON file, while ***load_configuration()*** loads the settings from the JSON file or creates it with default settings if it doesn't exist. Additionally, a ***validate_configuration()*** function makes sure all necessary keys are in the configuration dictionary, keeping things consistent. By changing these settings, it is aimed

to improve the model's performance and made it work well in different computational environments. You can find these adjustments in a file called ***"configuration.py"*** in the nnunet 'nnunetv2' directory. These changes greatly improve training and tuning.

```
ANISO_THRESHOLD = 3
default_num_processes = 8 if 'nnUNet_def_n_proc' not in os.environ else int(os.environ['nnUNet_def_n_proc'])
default_n_proc_DA = 2
conv_kernel_sizes = [(3, 3, 3), (3, 3, 3), (2, 2, 2)]
```

Figure 24 - some configuration adjustments specially (n_proc_DA).

Supported File Formats by nnU-net.

Reader	Supported File Formats	Additional Information
NaturalImage2DIO	.png, .bmp, .tif	
NibabelIO	.nii.gz, .nrrd, .mha	
NibabelIOWithReorient	.nii.gz, .nrrd, .mha	This reader will reorient images to RAS!
SimpleITKIO	.nii.gz, .nrrd, .mha	
Tiff3DIO	.tif, .tiff	3D tif images

Our dataset after unzipping it is in a '**.nii.gz**' format

Datasets must be in the **nnUNet_raw folder** (which you either define when installing nnU-Net or export/set every time you intend to run nnU-Net commands!). Datasets are associated with a dataset ID, a three-digit integer, and a dataset name (which you can freely choose): For example, in our case "**Dataset101_HepVes**" [43].

Within each dataset folder, the following structure is expected:

- Dataset001_HepaticVessels/
- ├── dataset. Json
- ├── imagesTr
- ├── imagesTs -> optional (included)
- └── labelsTr
- In our case we have added an additional folder in this directory i.e.,
- └── labelsTs

In this section, we created several folders to keep different parts of the project organized. These folders are:

- **imagesTr:** for training images
- **labelsTr:** for training labels
- **imagesTs:** for testing images
- **labelsTs:** for testing labels

We also made two main folders:

- **nnunet_raw:** to hold the four sub-folders.
- **nnunet_results:** to store the training results of models.

Additionally, we set up a folder named **'nnunet_preprocessed'** for preprocessed data. This organized structure helps manage and access the different datasets and outputs efficiently throughout the project.

```
task_name = 'Dataset101_HepVes' #task name
nnunet_dir = "nnUNet/nnunetv2/nnUNet_raw"
task_folder_name = os.path.join(nnunet_dir,task_name)
train_image_dir = os.path.join(task_folder_name,'imagesTr')
train_label_dir = os.path.join(task_folder_name,'labelsTr')
test_dir = os.path.join(task_folder_name,'imagesTs')
test_label_dir = os.path.join(task_folder_name,'labelsTs')
main_dir = os.path.join(base_dir,'nnUNet/nnunetv2')
```

```
make_if_dont_exist(task_folder_name,overwrite = False)
make_if_dont_exist(os.path.join(main_dir,'nnUNet_results'))
make_if_dont_exist(os.path.join(main_dir,'nnUNet_preprocessed'))
make_if_dont_exist(os.path.join(main_dir,'nnUNet_preprocessed_test'))
```

Figure 25 - Directories Creation and Specification.

nnU-Net uses specific environment variables to easily find raw data, preprocessed data, and trained models. There are three main environment variables:

- **nnunet_raw:** for the folder with the raw hepatic vessels data
- **nnunet_preprocessed:** for the folder with the preprocessed hepatic vessels data
- **nnunet_results:** for the folder where the results and trained models are stored.

These variables make it simple to organize and access important data and outputs during the nnU-Net workflow.

```
os.environ['nnUNet_raw'] = os.path.join(main_dir,'nnUNet_raw')
os.environ['nnUNet_preprocessed'] = os.path.join(main_dir,'nnUNet_preprocessed')
os.environ['nnUNet_preprocessed'] = os.path.join(main_dir,'nnUNet_preprocessed_test')
os.environ['nnUNet_results'] = os.path.join(main_dir,'nnUNet_results')
os.chdir(base_dir)
```

Figure 26 -Environment Variables.

Train test Split as 260 for train and 43 for the test (~85% training and ~15 testing -> this split allows the model to learn from a large portion of the data ‘training set’ while still leaving enough data for evaluating its performance ‘testing set’ and it is made sure that the split is representative of the dataset to avoid overfitting, 5 folds cross validation were used later on). This data splitting

has been done manually, after splitting the dataset, the Json file has been updated to update training and testing file paths.


```
"channel_names": {
    "0": "CT"
},
"description": "Hepatic Vessels and Tumour Segmentation",
"file_ending": ".nii.gz",
"labels": {
    "Tumour": 2,
    "Vessel": 1,
    "background": 0
},
"licence": "CC-BY-SA 4.0",
"name": "Dataset101_HepVes",
"numTest": 43,
"numTraining": 260,
"reference": "Memorial Sloan Kettering Cancer Center",
"release": "0.0",
"tensorImageSize": "3D",
```


Figure 27 - dataset updated.

3.2.1.2 Data Preprocessing and Planning

When the dataset is processed, nnU-Net automatically creates a dataset fingerprint, which includes details like image sizes, voxel spacings, and intensity information. This helps create three different U-Net configurations, each suited for its specific version of the preprocessed dataset. Using the command **nnUNetv2_plan_and_preprocess**, a special subfolder is created in the **nnUNet_preprocessed** directory, named after the dataset. After running the command, important files like dataset_fingerprint.json and nnUNetPlans.json are generated, along with subfolders containing preprocessed data for the different U-Net configurations.

Four planning techniques were tested carefully to find the best one with following commands:

- **!nnUNetv2_plan_and_preprocess -d 101 --verify_dataset_integrity**
- **nnUNetv2_plan_experiment -d 3 -pl nnUNetPlannerResEncM -gpu_memory_target 50 -overwrite_plans_name nnUNetResEncUNetPlans_50G**
- **!nnUNet_train 3d_fullres nnUNetTrainerV2 101 0**
- **!python experiment_planning/plan_and_preprocess_api.py -t 101 --verify_dataset_integrity**

Each method's efficiency, accuracy, and adaptability was evaluated. While each had its pros and cons, it was found that **!nnUNetv2_plan_and_preprocess -d 101 --verify_dataset_integrity** was the best choice. This method provided detailed planning and thorough dataset integrity verification, which is crucial for reliable and consistent experimental results. Its detailed insights and meticulous validation process gave me confidence in the planning phase, setting a solid foundation for my modeling and training efforts.

```
# Plan Method (1)
!nnUNet_train 3d_fullres nnUNetTrainerV2 101 0
# Plan Method (2) -> Best in our case
!nnUNetv2_plan_and_preprocess -d 101 --verify_dataset_integrity
# Plan Method (3)
!python experiment_planning/plan_and_preprocess_api.py -t 101 --verify_dataset_integrity
# Plan Method (4)
!nnUNetv2_plan_experiment -d 3 -pl nnUNetPlannerResEncM -gpu_memory_target 50 -overwrite_plans_name nnUNetResEncUNetPlans_50G
```

Figure 28 - Experiment planning and preprocessing.

In our CT dataset, our pipeline incorporates a group of instructional rules. The purpose is to get **data-dependent hyper-parameters**. These rules include crucial aspects such as the **image size** before and after cropping, ensuring consistency in the number of voxels per spatial dimension. Additionally, it is meticulously consider image spacing, capturing the physical size of each voxel, to maintain accuracy in spatial representation. Utilizing metadata, particularly modalities such as CT or MRI, aids in distinguishing between different imaging techniques and adapting the pipeline accordingly. Furthermore, the number of classes across all images and the complete number of training cases, essential metrics guiding the segmentation approach and model training process are assessed. This comprehensive approach ensures robustness and adaptability to the unique characteristics of the CT dataset, facilitating accurate and reliable segmentation results.

```
{
   "foreground_intensity_properties_per_channel": {
      "0": {
         "max": 345.0,
         "mean": 113.6622314453125,
         "median": 122.0,
         "min": -27.0,
         "percentile_00_5": 2.0,
         "percentile_99_5": 210.0,
         "std": 49.65023422241211
      }
   },
   "median_relative_size_after_cropping": 1.0,
   "shapes_after_crop": [
      [
         50,
         512,
         512
      ] ....
   ],
   "spacings": [
      [
         5.0,
         0.9746090173721313,
         0.9746090173721313
      ], ...    ]
}
```


Figure 29 - Dataset Fingerprints

In our data preprocessing pipeline, **intensity normalization** is crucial for maintaining consistent and standardized input across different types of medical images. nnUNet uses **Z-scoring** to normalize intensities for most modalities, except for CT scans. For CT images, nnUNet applies a **global normalization** approach using the **0.5** and **99.5** percentiles of foreground voxels to clip outliers and preserve meaningful intensity distributions.

Resampling is also important to handle the natural variations in medical imaging data. All images are resampled to a uniform target spacing using interpolation methods like third-order spline, linear, or nearest neighbor in nnUNet. This helps to ensure uniformity and smoother processing. The architecture of nnUNet's network, **batch size**, and **patch size** are adjusted based on the average

image size after resampling and the target spacing. This optimization helps nnUNet make the best use of computational resources and adapt to different image characteristics.

The patch size is initialized to match the average image shape post-resampling in nnUNet, ensuring compatibility with subsequent processing steps. The architectural design is further refined by assessing the number of operations needed for downsampling in nnUNet, ensuring effective feature extraction while preserving spatial information. Default kernel sizes for convolutions are set to **3D U-Net** and **2D U-Net** as **3×3×3** and **3×3**, respectively, optimizing the design for efficient and accurate segmentation.


```
"configurations": {
    "2d": {
        "data_identifier": "nnUNetPlans_2d",
        "preprocessor_name": "DefaultPreprocessor",
        "batch_size": 12,
        "patch_size": [
            512,
            512
        ],
        "median_image_size_in_voxels": [
            512.0,
            512.0
        ],
        "spacing": [
            0.8515620231628418,
            0.8515620231628418
        ],
        "normalization_schemes": [
            "CTNormalization"
        ],
        "use_mask_for_norm": [
            false
        ],
        "UNet_class_name": "PlainConvUNet",
        "UNet_base_num_features": 32,
        "n_conv_per_stage_encoder": [
            2,
            2,
            2,
            2,
            2,
            2,
            2,
            2
        ],
        "n_conv_per_stage_decoder": [
            2,
            2,
            2,
            2,
            2,
            2,
            2
        ],
        "num_pool_per_axis": [
            7,
            7
        ],
```


Figure 30 - Parameters (nnUnet Plans File)

3.2.1.3 The Model (Training – Tuning - Prediction)

After thorough examination and analysis, it has been determined that, for our dataset, the nnUNet **2D** and nnUNet **3D LowRes** approaches are preferred over the others. The nnUNet 2D method processes images slice by slice, offering computational efficiency that aligns well with our dataset's characteristics, despite potentially losing some 3D contextual information. Similarly, the nnUNet 3D LowRes technique operates at reduced resolutions during both training and inference, hitting

a balance between preserving essential details and computational efficiency, which makes it well-suited for our dataset's requirements.

- **Model Training:**

In My pursuit of optimal model performance, I've adopted a dual training strategy. Firstly, each model variant is trained for **20 epochs** to quickly explore initial learning dynamics and performance trends. This approach provides insights into the model's adaptability and convergence patterns early on. Secondly, training configuration was extended to **100 epochs** to achieve deeper knowledge and refine parameters.

Throughout both training phases, nnU-Net uses a **5-fold cross-validation** method across training cases. This ensures thorough evaluation and model selection, allowing nnU-Net to assess the performance of each configuration and identify the most suitable one for our segmentation problem. Therefore, the 5-fold cross-validation is integrated into both the 20-epoch and **100-epoch** training cycles, ensuring strict evaluation and informed decision-making throughout the training process.

- **20 Epochs Approach:**

nnU-Net was tested to see how well it could segment hepatic vessels in both **2D** and **3D**. The experiment ran for **20 epochs** to check its performance over time. Each epoch was a complete pass through all our data. It was quite a process, as the accuracy and convergence metrics were closely monitored at each step to see how well the model learned to segment hepatic vessels.

Over the 20 epochs, it is noticed both short-term changes and long-term patterns, which helped in understanding how well nnU-Net worked for this task. By the end of the experiment gained valuable insights into how effectively nnU-Net handles hepatic vessel segmentation in both 2D and 3D scenarios.

```
os.chdir(main_dir)

!nnUNetv2_train 101 2d "all" -tr "nnUNetTrainer_20epochs"
!nnUNetv2_train 101 3d_lowres "all" -tr "nnUNetTrainer_20epochs"

os.chdir(base_dir)
```

Figure 31 - 20 epochs training.

2D Configuration:

Throughout the 20 epochs of training for the 2D hepatic vessel segmentation task, it has been observed that there is a clear improvement in nnU-Net's adaptability and learning dynamics.

- **Firstly,** analyzing the learning rate chart over epochs showed a consistent downward trend, indicating the model's gradual adjustment to the complexities of the dataset. This decrease in learning rate allowed the model to better understand the detailed features in the hepatic vessel images, leading to more accurate segmentation.
- **Secondly,** the epoch duration graph demonstrated a noticeable trend: a steady decrease in epoch duration as training progressed. This reduction reflects the model's improved efficiency in processing and learning from the data, showing its ability to optimize computational resources over time.
- **Lastly,** monitoring the pseudo dice scores and loss over epochs provided insights into the model's convergence and performance. Here, a consistent improvement in pseudo dice scores alongside a corresponding decrease in loss values is observed, showing that the model achieved higher segmentation accuracy and reduced errors throughout the training period.

  This detailed analysis of the training progress highlights nnU-Net's effectiveness in learning and adapting to the challenges of 2D hepatic vessel segmentation.

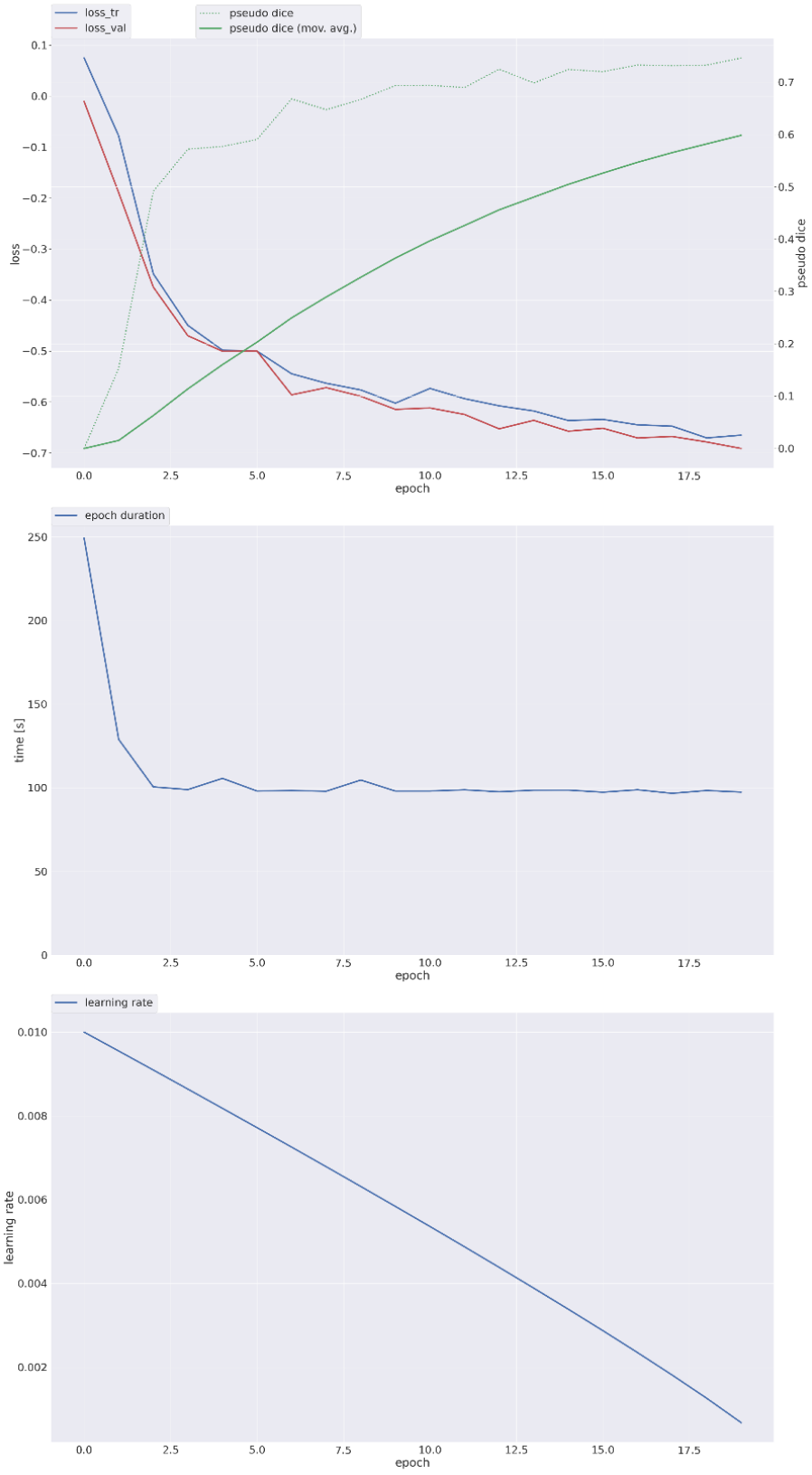


Figure 32 - progress of the 20 epochs training (2d).

In the experiment, five random samples were used to assess how well nnU-Net performs in segmenting hepatic vessels. After training was complete, test images alongside their actual labels and the predicted labels generated by nnU-Net were examined. The visual inspection revealed impressive results: the predicted labels closely matched the actual annotations for some of the sampled images. This alignment between the predicted and actual labels confirmed nnU-Net's ability to accurately segment hepatic vessels from complex imaging data.

By showing the segmentation results visually, the approach not only helped in assessing the quality of the segmentation but also boosted my confidence in nnU-Net's predictions. These diagrams are valuable for demonstrating nnU-Net's effectiveness in hepatic vessel segmentation tasks and providing valuable insights into its performance on real-world imaging data.

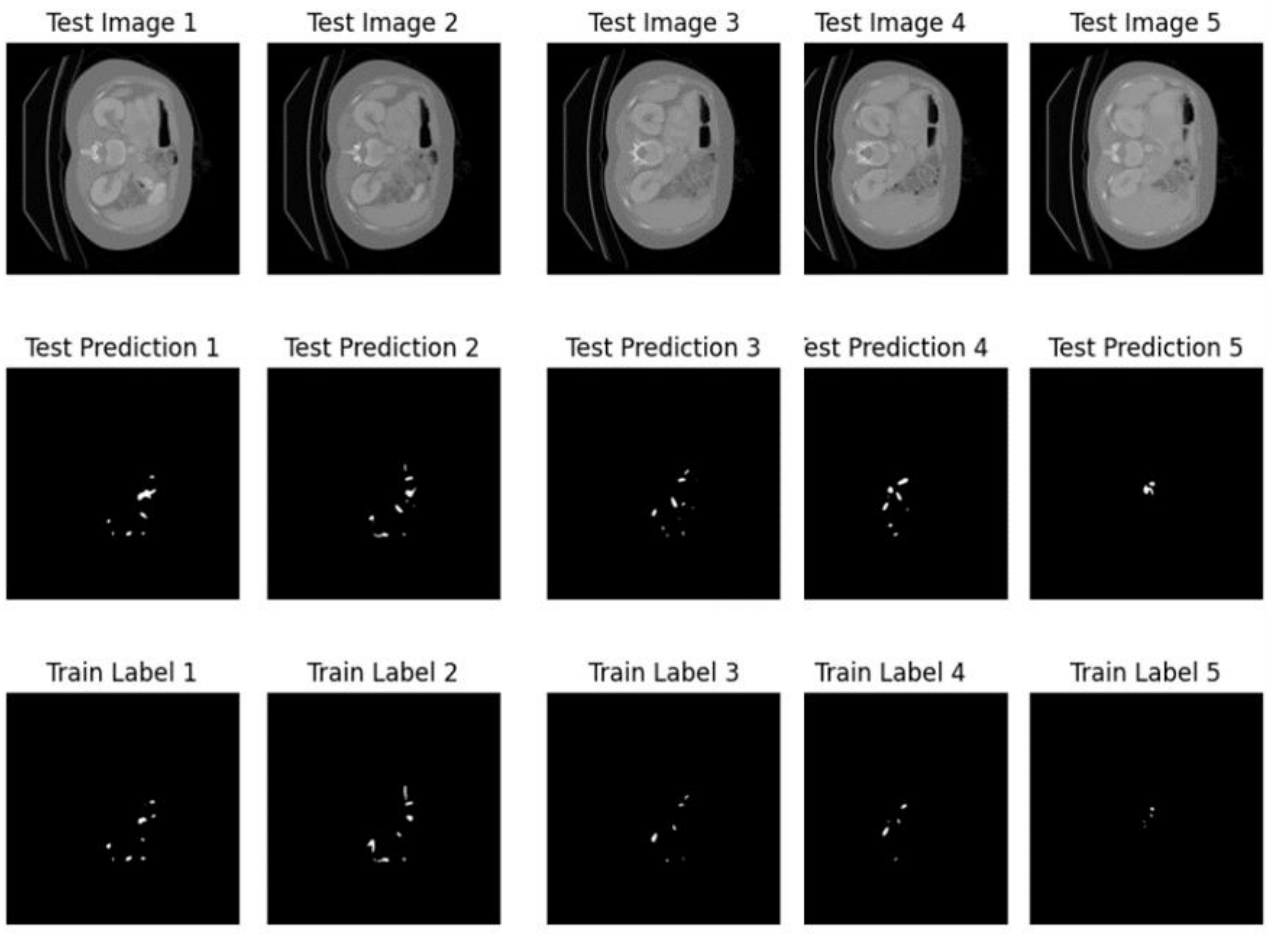


Figure 33 - 2d (20 epochs) predictions.

3d_lowRes configuration:

In the 3D_lowres segmentation task over 20 epochs, nnU-Net's adaptability and learning progress were noticed. The learning rate consistently dropped, showing the model's adjustment to the dataset's details and leading to more precise segmentation. The duration of each epoch also constantly decreased, indicating the model's improved efficiency in processing data over time. Monitoring the pseudo dice and loss metrics showed improved segmentation accuracy and fewer errors throughout training, highlighting nnU-Net's effectiveness in handling the complexities of 3D_lowres segmentation.

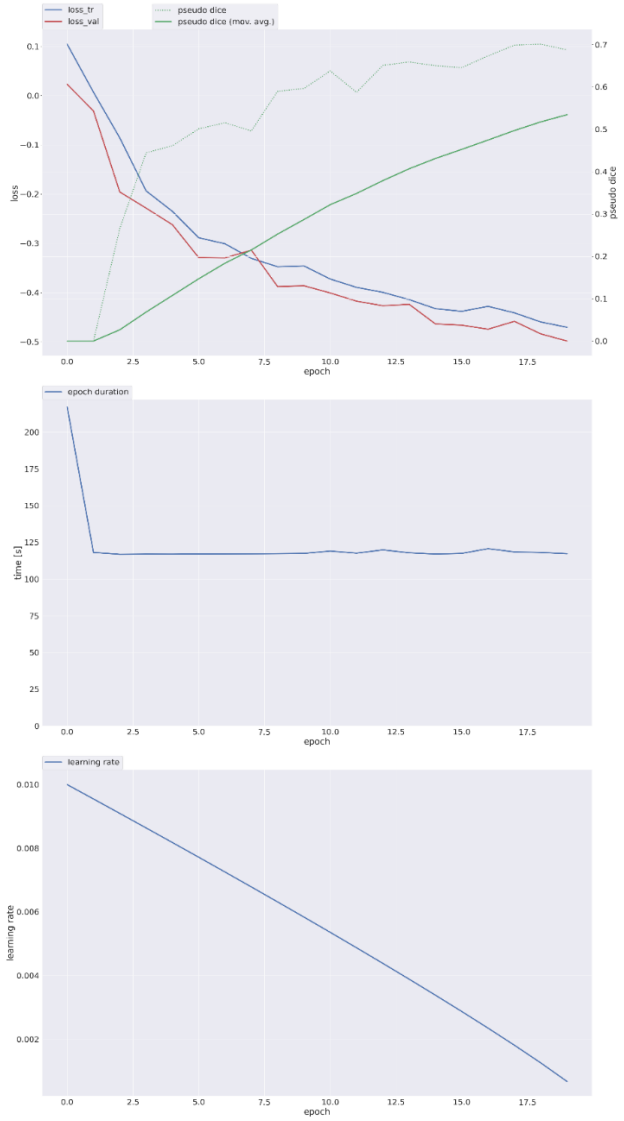


Figure 34 -progress of the 20 epochs training (3d lowRes).

nnU-Net's performance was assessed in 3D lowres segmentation using five random samples. Visual comparisons of test images, ground truth labels (vessels), and nnU-Net's predictions revealed strong alignment, confirming its accuracy. These visuals provide compelling evidence of nnU-Net's effectiveness in 3D lowres segmentation.

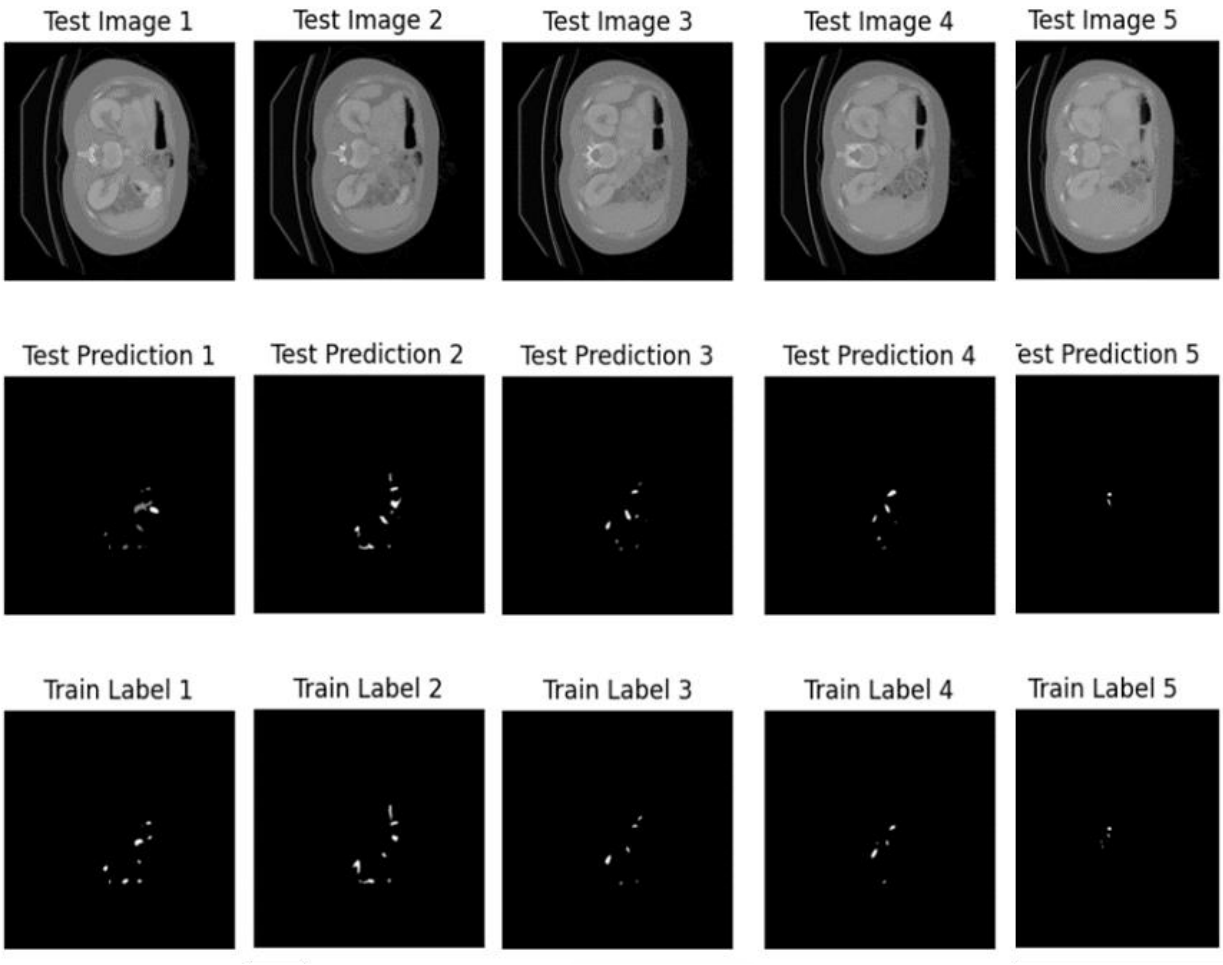


Figure 35 - 3d lowres predictions.

From my observations, in a low-resolution configuration over 20 epochs, 2D hepatic vessel segmentation performs better than the 3D approach. The two-dimensional information provides a more straightforward and efficient means of understanding vascular structures in the liver under these conditions. This simplicity allows nnU-Net to effectively capture essential details within the constraints of low-resolution imaging data, leading to more accurate segmentation results.

Moreover, visualizations of the predictions confirmed this finding, as the 2D predictions aligned more closely with the actual annotations compared to the 3D predictions. The 2D approach bypasses the complexities associated with volumetric characteristics, offering a more precise view of the hepatic vasculature in low-resolution settings.

As a result, nnU-Net demonstrates superior performance in 2D hepatic vessel segmentation under these specific conditions, highlighting the benefits of using two-dimensional information in medical image analysis when dealing with low-resolution data.

- **100 Epochs (Full Power)**

A detailed experiment to test nnU-Net's performance in hepatic vessel segmentation was conducted in different setups: 2D, 3D lower resolution, 3D full resolution, and a combined approach. The training to 100 epochs was extended to analyze the model's learning more deeply. Each epoch was a full pass through the dataset, allowing to closely monitor accuracy and convergence.

During this extended training, all changes and trends were carefully tracked to understand nnU-Net's adaptability and effectiveness in segmenting hepatic vessels. This long experiment gave valuable insights into nnU-Net's capabilities in both 2D and 3D scenarios.

```
os.chdir(main_dir)

!nnUNetv2_train 101 2d "all" -tr "nnUNetTrainer_100epochs"
!nnUNetv2_train 101 3d_lowres "all" -tr "nnUNetTrainer_100epochs"
!nnUNetv2_train 101 3d_fullers "all" -tr "nnUNetTrainer_100epochs"
!nnUNetv2_train 101 3d_cascade_fullers "all" -tr "nnUNetTrainer_100epochs"

os.chdir(base_dir)
```

Figure 36 - 100 epochs training.

2D Configuration Validation Results:

Metric	Mean 1	Mean 2	foreground_mean
Dice	0.6637428731583944	0.6748802876512052	0.6693115804047998
FN	5536.188118811881	6332.039603960396	5934.1138613861385
FP	6980.448844884489	6751.706270627063	6866.077557755776
IoU	0.5070978353362477	0.5666202591941703	0.536859047265209
TN	18246162.8679868	18206889.231023103	18226526.04950495
TP	13535.940594059406	52242.46864686468	32889.20462046204
n_pred	20516.389438943894	58994.17491749175	39755.28217821782
n_ref	19072.128712871287	58574.508250825085	38823.31848184818

Figure 37 - Validation Results of nnU-net 2D.

- **Dice Score**: The Dice score of Tumors (class 2) is 0.67 and the Dice score of Vessels (class 1) is 0.66, indicating moderate performance in terms of segmentation accuracy.
- **False Negatives (FN)**: There are around 6332 false negatives, which suggests a significant number of missed foreground pixels.
- **False Positives (FP)**: The false positive count is around 6752, indicating some over-segmentation.
- **IoU (Intersection over Union)**: IoU is around 0.57, indicating a moderate level of overlap between predicted and ground truth masks.

3D Full Resolution Configuration Validation Results:

Metric	Mean 1	Mean 2	foreground_mean
Dice	0.6535239315559608	0.7499720326359156	0.7017479820959382
FN	6233.40329218107	7154.843621399177	6694.123456790124
FP	6706.588477366256	9193.374485596707	7949.981481481482

IoU	0.4957264261100304	0.636517468020459	0.5661219470652448
TN	17088445.21399177	17036855.658436213	17062650.436213993
TP	12410.744855967077	60592.07407407407	36501.40946502057
n_pred	19117.333333333332	69785.44855967078	44451.390946502055
n_ref	18644.14814814815	67746.91769547325	43195.5329218107

Figure 38 - Validation Results of nnU-net 3D FullRes.

- **Dice Score**: Improved Dice score compared to 2D, reaching 0.75, indicating better segmentation performance.
- **FN and FP**: False negatives and false positives have reduced compared to the 2D configuration, indicating better segmentation.
- **IoU**: Shows significant improvement compared to 2D, reaching 0.64, indicating better overlap between predicted and ground truth masks.

3D Low Resolution Configuration Validation Results:

Metric	Mean 1	Mean 2	Foreground_mean
Dice	0.6739023909165712	0.7934443930970612	0.7336733920068161
FN	6105.020576131687	6039.411522633744	6072.216049382716
FP	5362.415637860082	8860.506172839507	7111.460905349794
IoU	0.5185196798919184	0.683250783337918	0.6008852316149182
TN	17089789.386831276	17037188.52674897	17063488.956790123
TP	12539.127572016461	61707.50617283951	37123.31687242798
n_pred	17901.543209876545	70568.01234567902	44234.77777777778
n_ref	18644.14814814815	67746.91769547325	43195.5329218107

Figure 39 - Validation Results of nnU-net 3D LowRes.

- **Dice Score**: Highest Dice score among all configurations, reaching 0.79, indicating the best segmentation performance.
- **FN and FP**: False negatives and false positives are lower compared to both 2D and 3D full resolution configurations.
- **IoU**: Also shows improvement compared to 2D, but slightly lower than the 3D full resolution configuration, indicating good overlap but not as high as in full resolution.

3D Full Resolution Cascade Configuration Validation Results:

Metric	Mean 1	Mean 2	Foreground_mean
Dice	0.6666395829342098	0.7632123794742334	0.7149259812042216
FN	5857.958847736625	5780.201646090535	5819.08024691358
FP	6414.374485596708	11432.77366255144	8923.574074074075
IoU	0.5106141716469698	0.6502411621798152	0.5804276669133925
TN	17088737.427983537	17034616.259259257	17061676.843621396
TP	12786.189300411523	61966.71604938272	37376.45267489712
n_pred	19200.56378600823	73399.48971193416	46300.0267489712
n_ref	18644.14814814815	67746.91769547325	43195.5329218107

Figure 40 -Validation Results of nnU-net 3D Cascade.

- **Dice Score:** Similar to the 3D full resolution configuration, indicating good segmentation performance.
- **FN and FP:** False negatives and false positives are relatively higher compared to the 3D low resolution configuration but lower than the 2D configuration.
- **IoU:** Slightly lower than the 3D full resolution configuration but higher than the 2D configuration, indicating decent overlap.

Summary:

- The **3D low resolution configuration** performs the best overall in terms of Dice score, indicating the highest segmentation accuracy.
- The **3D full resolution configuration** follows closely behind, showing good performance across metrics.
- The **3D full resolution cascade configuration** performs similarly to the full resolution configuration but with slightly higher false positives.
- The **2D configuration** consistently performs the worst across all metrics, indicating the lowest segmentation accuracy.

Overall, transitioning from 2D to 3D segmentation and utilizing higher resolution data leads to improved segmentation performance, with the 3D low resolution configuration being the most effective in this case.

Here are the Progress of the 100 epochs in all the 4 different configurations.

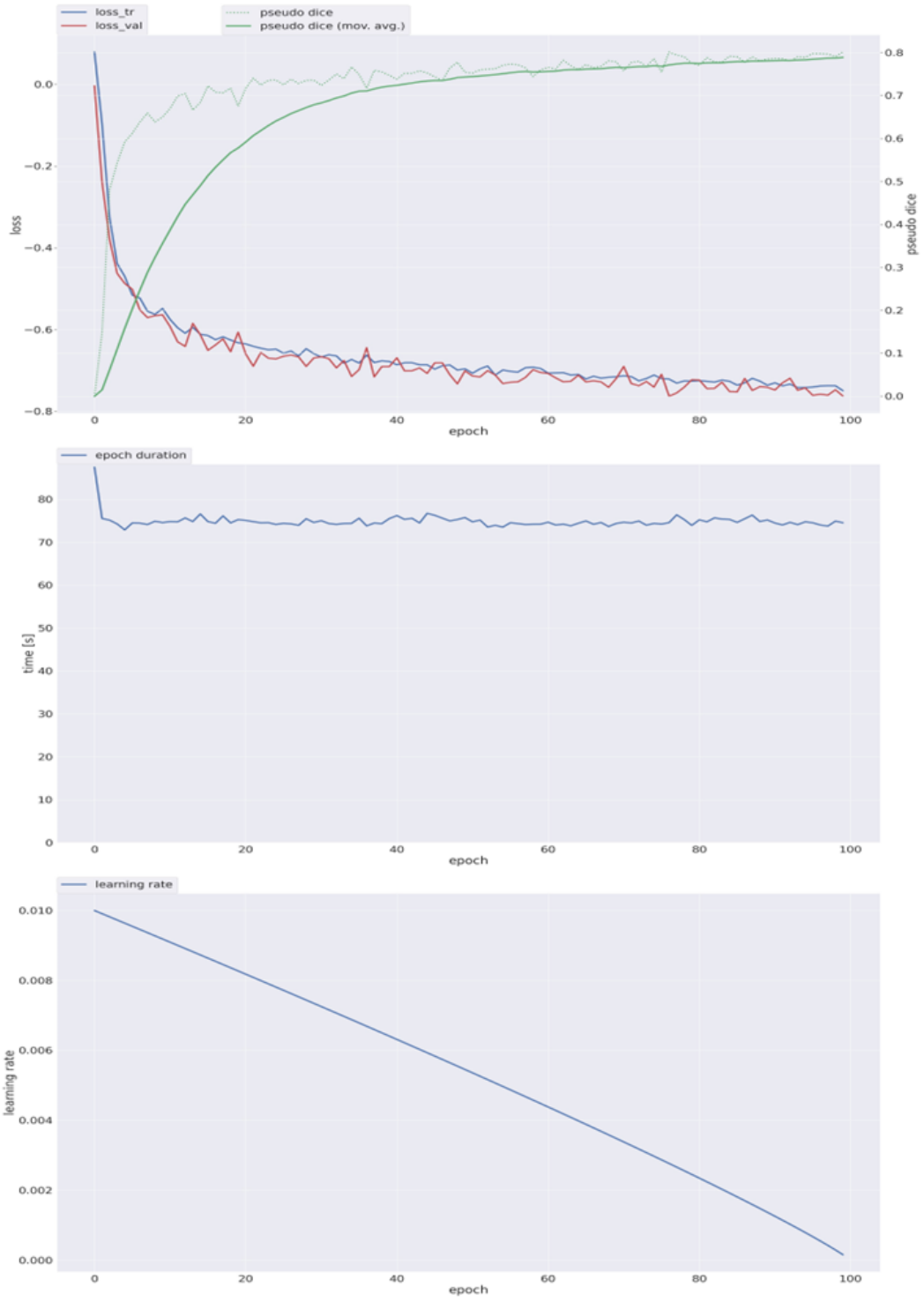


Figure 41 -- nnUnet 2D.

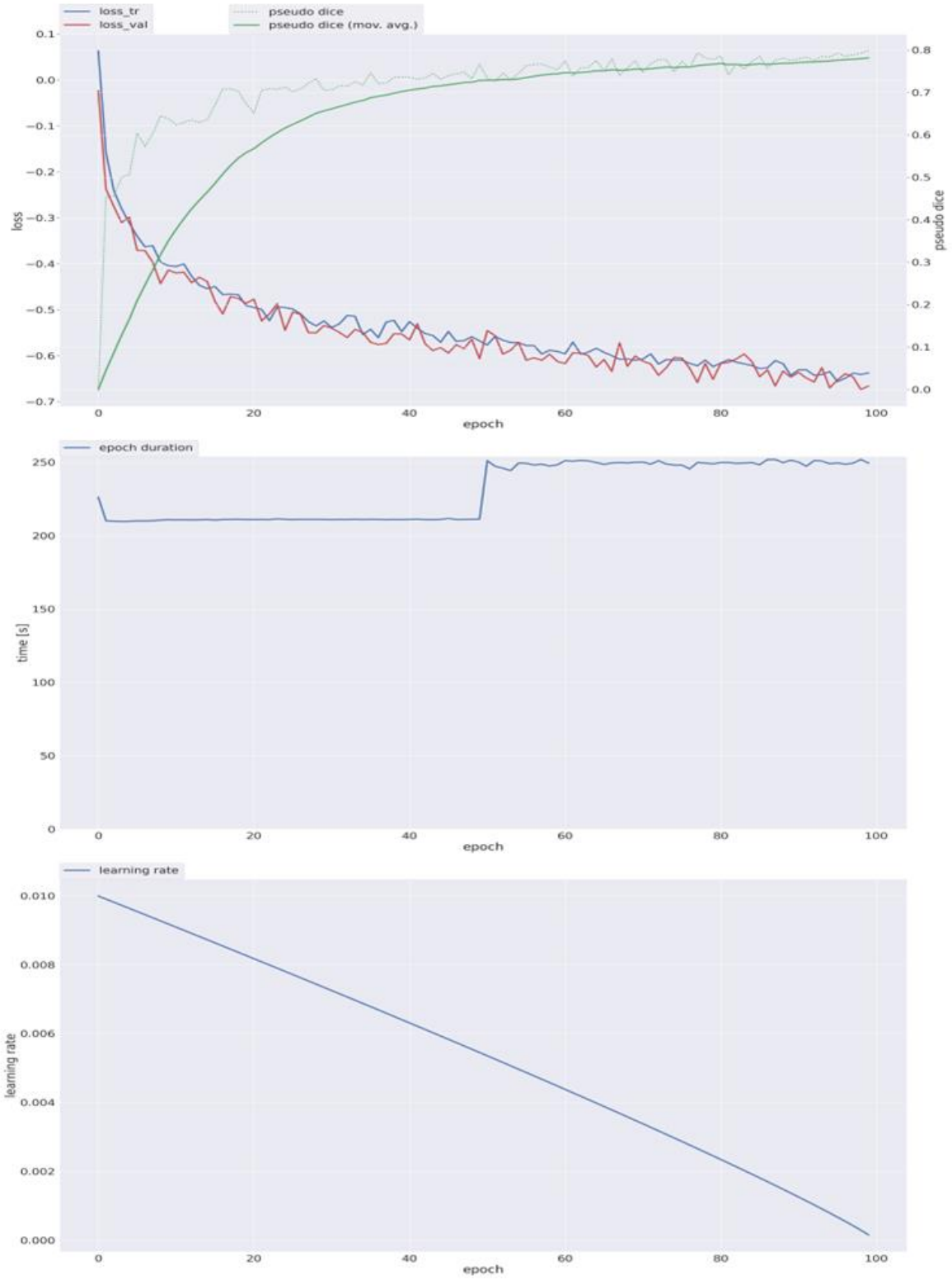


Figure 42 - nnUnet 3D LowRes.

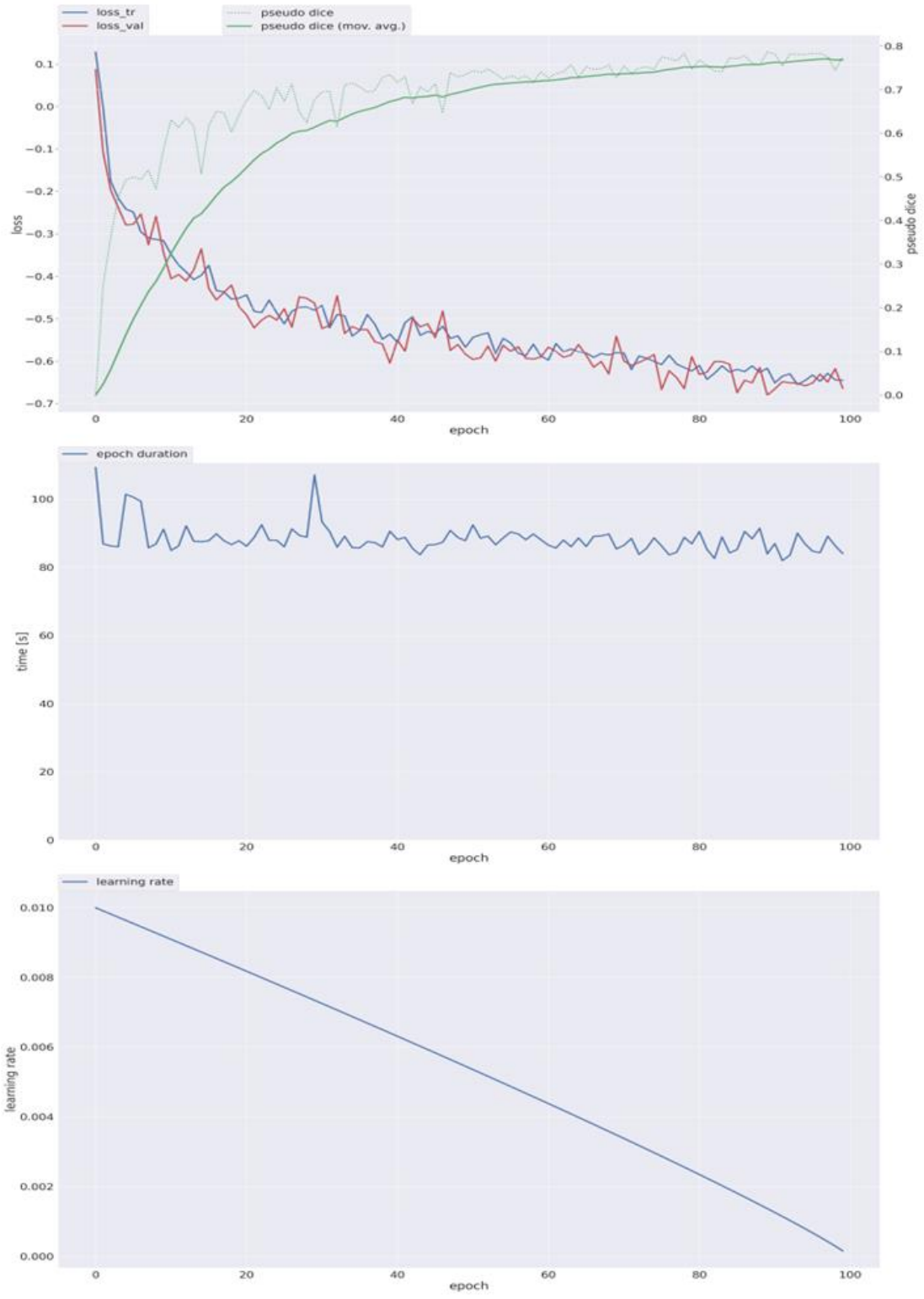


Figure 43 - nnUnet 3D FullRes.

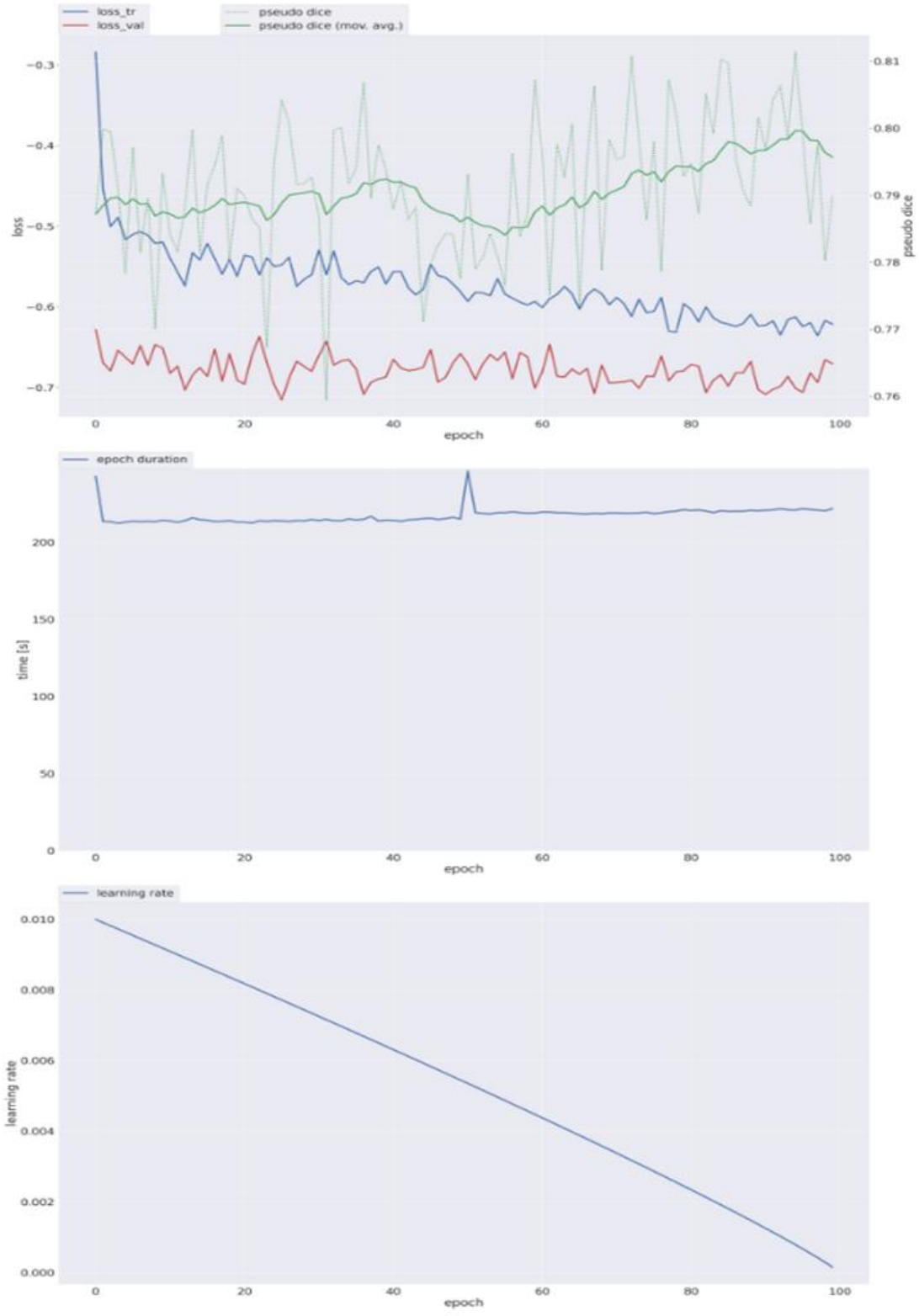


Figure 44 - - nnUnet Cascade fullRes.

Here are the Predictions of the 100 epochs of the three best configurations.

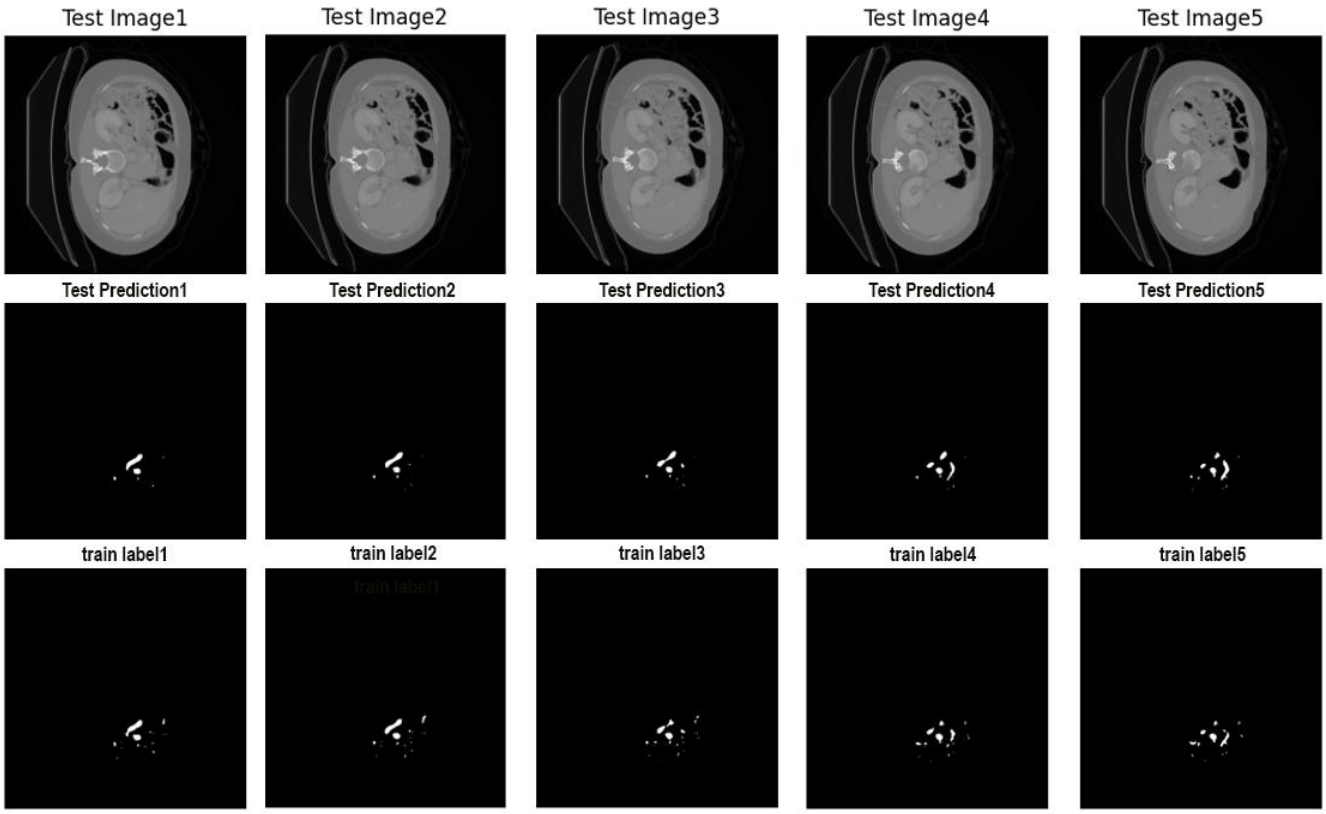


Figure 45 - nnunet 2D predictions.

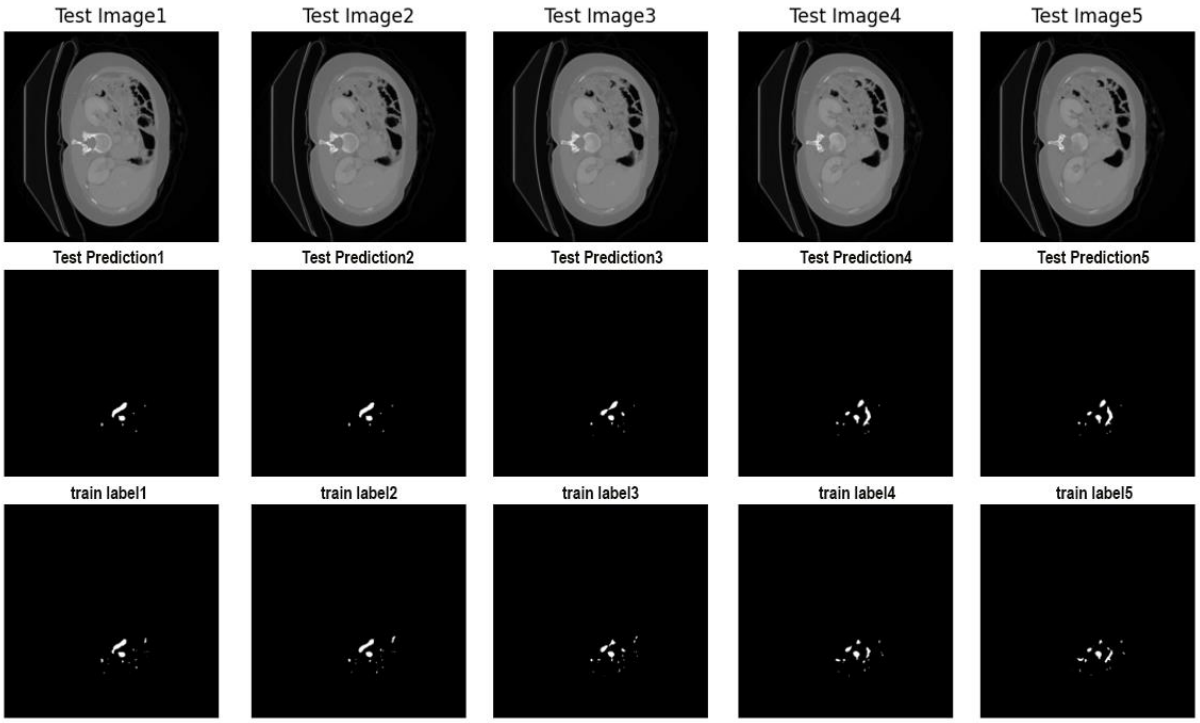


Figure 46 - nnunet 3D Fullres predictions.

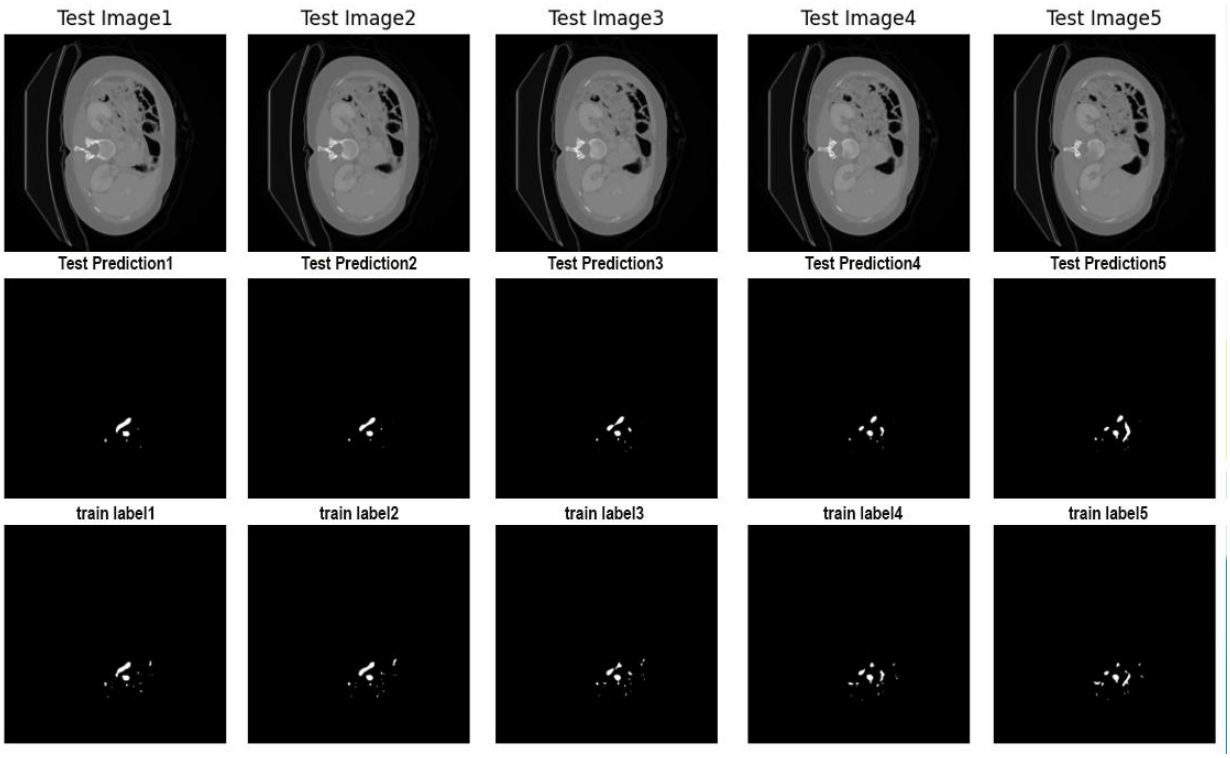


Figure 47 - nnunet 3D LowRes.

3.3 CNN (U-net) Traditional Technique

When working with CNN, there were two main options. The first option is to use preprocessed data from nnunet, which is organized in specific folders and files with extensions like **.npy**, .**npz,** and **seg.npy.** [46]. This will be discussed in an upcoming section.

The second option is to build the CNN architecture from scratch, using standard organization and preprocessing methods. By not relying on nnunet's specialized preprocessing, potential issues were avoided with its specific approach. Using more common methods makes our approach more flexible and compatible with various datasets beyond nnunet's scope. Starting from scratch also lets us show how well nnunet works with our smaller dataset. The next sections will explain these techniques in more detail and discuss why these choices were made [47].

3.3.1 Data Preparation

Once **Google Drive** was connected to **Google Colab** and installed the necessary libraries, next step was to organize the data into a new directory named "**cnn_vessels**." The data folders and files within this isolated directory were carefully arranged.

Next, the data was loaded, which included training and testing images *(imagesTr*, *imagesTs*), and aligned them accurately with their respective labels (*labelsTr*, *labelsTs*). This alignment was crucial to ensure that each image matched accurately with its corresponding label, establishing the foundation for the next steps.

To streamline upcoming tasks and prepare for what was ahead, we standardized the target shape to *(256, 256, 256).* This standardized shape not only made later processes easier but also ensured consistency across the dataset.

With everything set up, we proceeded with our initial visualization, marking the beginning of exploration and analysis.

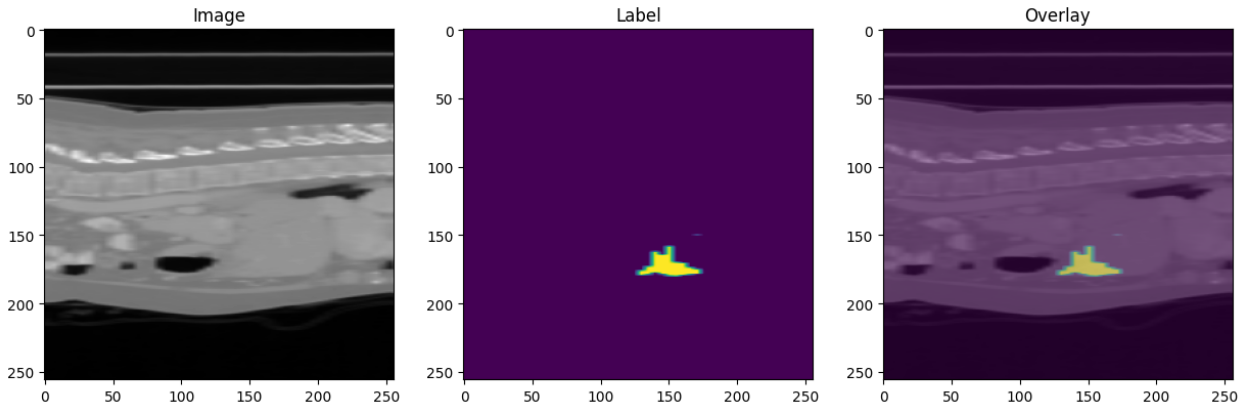


Figure 48 - Stepp 1, Data Visualization.

3.3.2 Intensity Normalization

To start preprocessing image data, intensity normalization was first step. This important process helped to standardize the intensity levels across all images, ensuring consistency and reducing variations. Normalizing the intensity aimed to improve the comparability and reliability of our dataset, establishing a strong foundation for further analyses and model training.

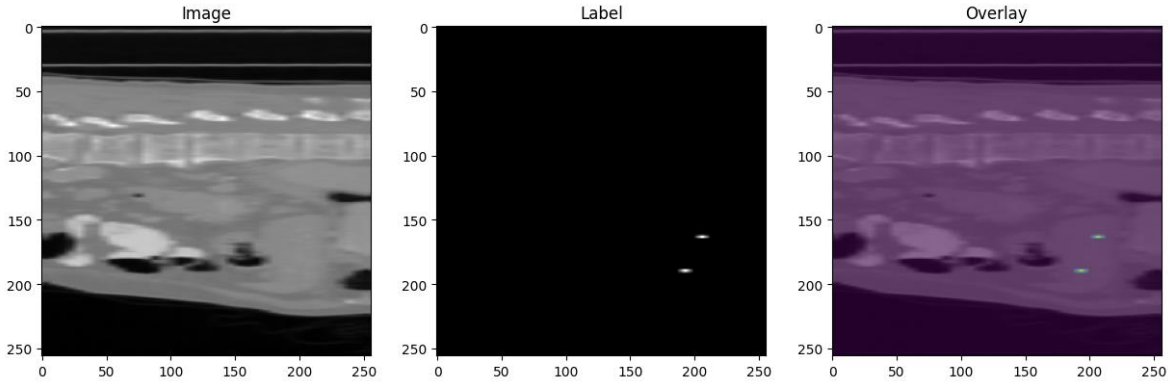


Figure 49 - Step 2, Intensity Normalization.

3.3.3 Noise Reduction

After normalizing intensity, next step in the data preprocessing pipeline was to apply noise reduction techniques to both images and labels. Noise reduction was crucial for refining the dataset by removing unwanted artifacts and irregularities that could affect the accuracy of later analyses and model training. By using noise reduction algorithms, the goal was to improve the clarity and fidelity of both the images and their labels, therefore enhancing the overall quality of the dataset.

This critical preprocessing step ensured that the data was clean, reliable, and prepared for further processing and analysis.

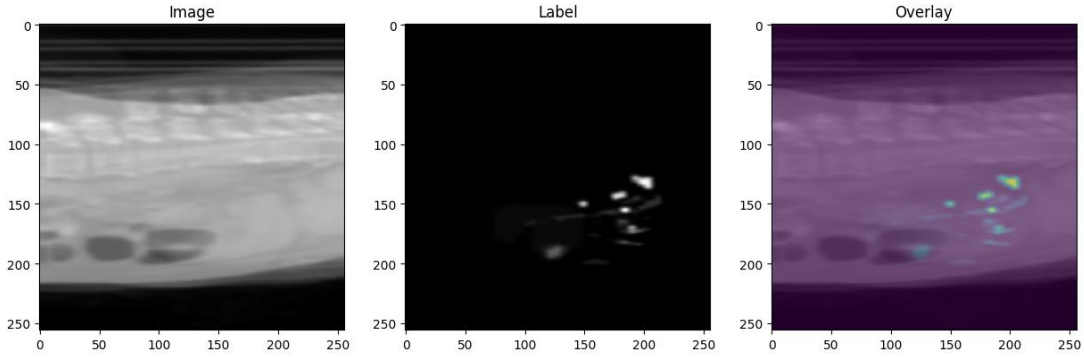


Figure 50 - Step 3, Noise Reduction.

3.3.4 Data Resampling

Continuing our data preprocessing efforts, the third step involved resampling the data to achieve consistent voxel dimensions across all images and labels. Resampling was critical to standardize the spatial resolution of the dataset, ensuring uniformity and facilitating integration during subsequent analyses and model training. By resampling the data, we aimed to resolve any inconsistencies in voxel dimensions, therefore improving the compatibility and reliability of our dataset. This essential preprocessing step set the stage for more accurate explanations, enabling smoother workflows and enhancing the performance of our models.

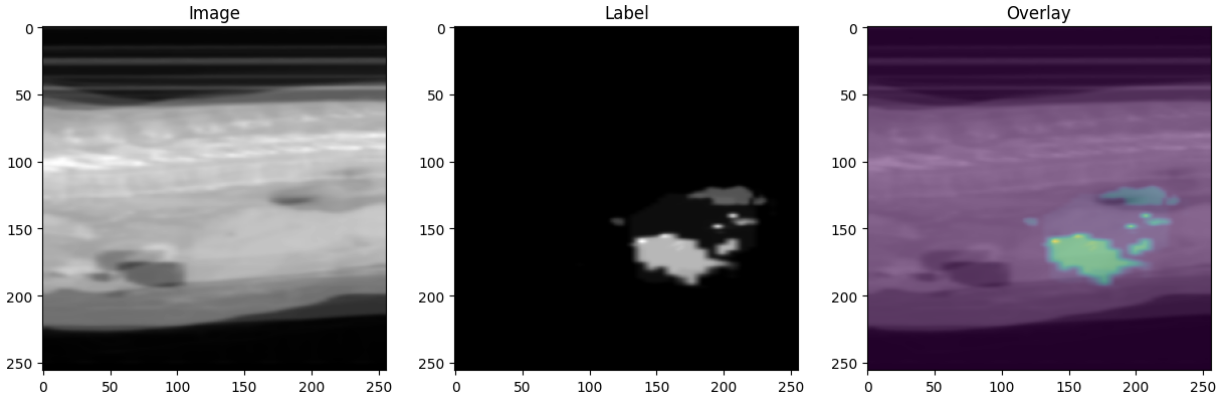


Figure 51 - Step 4, Resampling the data.

3.3.5 Data Augmentation

Because our training dataset was small, with **only 260 images and labels**, we needed more data to make our model work better and handle different situations. To do this, we used data augmentation with the **ImageDataGenerator** module from TensorFlow's Keras library. This tool helped us create **1000 more images** (Evaluating the model's performance after the augmentation. Didn't show significant improvement, 1000 images should be sufficient in our case or at least gives significant better results but it still needs more real data depending on my observation and testing), increasing the size of our training dataset. Data augmentation added variations like rotations, shifts, flips, and zooms to the images we already had. This made our dataset more diverse and helped our model learn better, reducing the risk of overfitting and improving its performance on new, unseen data.

3.3.6 Model (Build and Train)

This technique uses a modified UNet model designed specifically for our segmentation task. I've made some key improvements, like adding batch normalization and dropout, and enhancing the convolutional layers. The architecture is set up to handle 3D data accurately.

It starts with the encoder, which processes the data using convolutions and pooling to capture important features. At the bottleneck, we extract the most critical information. Then, the decoder reconstructs these features back to the original size for detailed segmentation. C**ustom loss functions—Dice coefficient and its loss counterpart—optimized** was used for our needs.

Additionally, the module can easily load our resampled images and labels, preparing our data for training and testing. With **TensorFlow** and **NumPy**, this setup is ready for segmentation tasks, especially in areas like medical imaging or volumetric analysis.

Two different model complexities were tested. In one case, the data was reshaped to *(128,128,128)* to avoid memory errors.

```
Epoch 1/10
65/65 [==============================] - ETA: 0s - loss: -0.0153 - dice_coef: 1.0153/usr/local/lib/python3.10/dist-packages/keras/src/engi
  saving_api.save_model(
65/65 [==============================] - 78s 778ms/step - loss: -0.0153 - dice_coef: 1.0153 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 2/10
65/65 [==============================] - 34s 531ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 3/10
65/65 [==============================] - 34s 530ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 4/10
65/65 [==============================] - 35s 532ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 5/10
65/65 [==============================] - 35s 532ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 6/10
65/65 [==============================] - 34s 531ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 7/10
65/65 [==============================] - 35s 532ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 8/10
65/65 [==============================] - 34s 530ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 9/10
65/65 [==============================] - 35s 533ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
Epoch 10/10
65/65 [==============================] - 35s 532ms/step - loss: -0.0680 - dice_coef: 1.0680 - val_loss: -0.0556 - val_dice_coef: 1.0556
```

Figure 52 - Model training (model no.1)

```
Epoch 1/10
227/227 [==============================] - ETA: 0s - loss: 0.0920 - dice_coef: 0.9080/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: U
  saving_api.save_model(
227/227 [==============================] - 224s 846ms/step - loss: 0.0920 - dice_coef: 0.9080 - val_loss: -0.0361 - val_dice_coef: 1.0361 - lr: 1.0000e-04
Epoch 2/10
227/227 [==============================] - 172s 757ms/step - loss: 0.0249 - dice_coef: 0.9751 - val_loss: -0.0047 - val_dice_coef: 1.0045 - lr: 1.0000e-04
Epoch 3/10
227/227 [==============================] - 172s 756ms/step - loss: 0.0065 - dice_coef: 0.9935 - val_loss: -0.0119 - val_dice_coef: 1.0119 - lr: 1.0000e-04
Epoch 4/10
227/227 [==============================] - 172s 757ms/step - loss: -0.0057 - dice_coef: 1.0057 - val_loss: -0.0188 - val_dice_coef: 1.0188 - lr: 1.0000e-04
Epoch 5/10
227/227 [==============================] - 172s 757ms/step - loss: -0.0116 - dice_coef: 1.0116 - val_loss: -0.0057 - val_dice_coef: 1.0057 - lr: 1.0000e-05
Epoch 6/10
227/227 [==============================] - 172s 757ms/step - loss: -0.0126 - dice_coef: 1.0126 - val_loss: -0.0037 - val_dice_coef: 1.0037 - lr: 1.0000e-05
```

Figure 53 - Model training (model no.2)

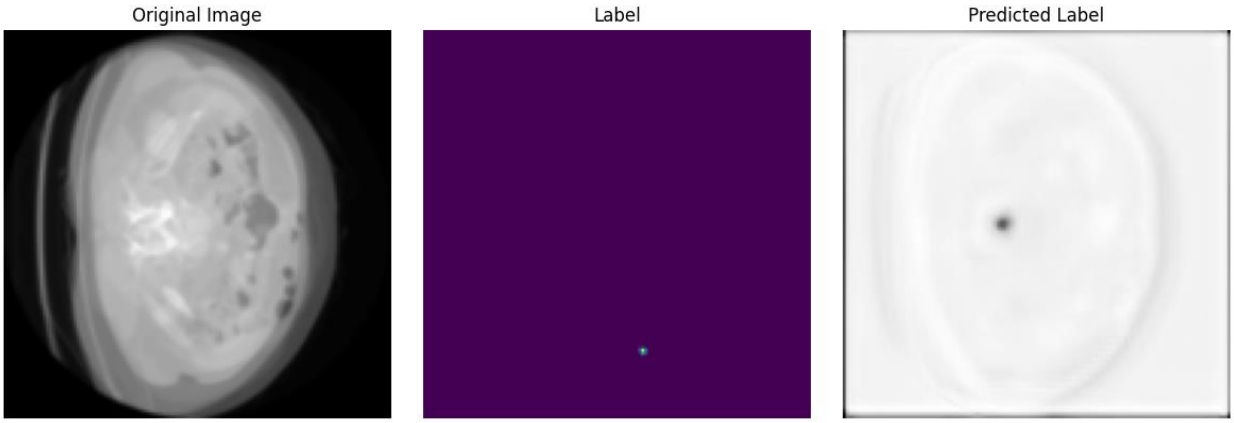


Figure 54 – Model 1 prediction.

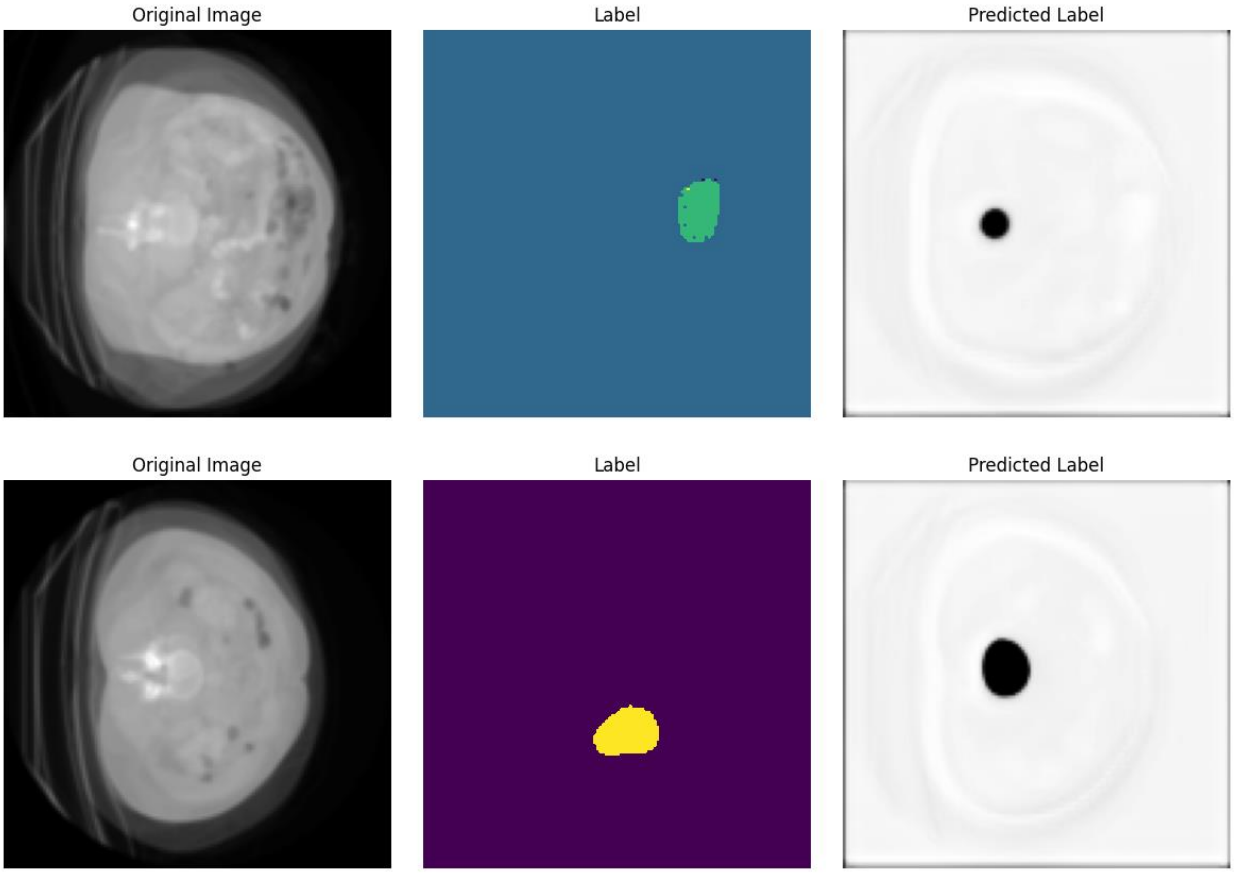


Figure 55 - Model 2 prediction.

We chose to train for 10 epochs at first, but despite trying to prevent over-fitting, the model still has problems. This is mainly because our dataset is small, even after adding more data through augmentation. These limitations are clear when we look at the predictions.

3.4 CNN (U-net) Utilizing nnU-net Preprocessed Data Technique

To overcome the modest results from the previous attempt of starting the data preprocessing from scratch using traditional techniques before building the model, it was decided to build the model on top of the preprocessed data that comes from the nnU-net tool as it uses advanced preprocessing techniques and with larger images shapes as well.

3.4.1 nnU-net Preprocessed Data (CT images)

nnU-net preprocessed data is excellent for medical imaging, especially for CT scans. Known for its advanced techniques, nnU-net uses powerful image normalization algorithms to standardize

image intensities across different datasets, ensuring consistent and reliable analysis. It also has state-of-the-art data augmentation strategies that create diverse training samples, which are essential for strong model training.

Additionally, nnU-net excels in image registration and alignment, making it easy to integrate multi-modal imaging data for comprehensive analysis. It effectively handles noise reduction and artifact suppression, delivering clear, artifact-free images. This high-quality preprocessing gives researchers and clinicians exceptional diagnostic precision and insight.

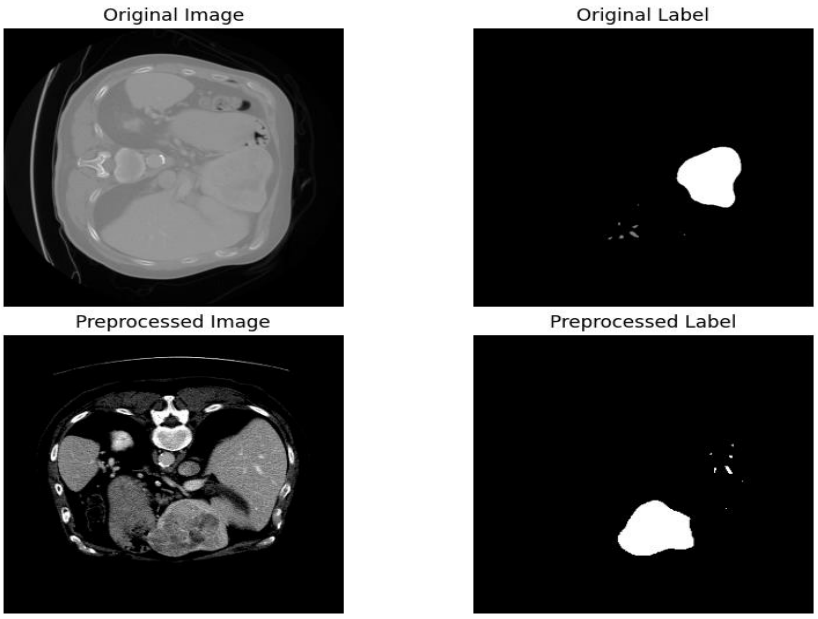


Figure 56 - nnU-net Preprocessing Results.

3.4.2 Data Preparation

After preprocessing, the focus was on preparing the data for building the model.

3.4.2.1 Evaluating Min and Max Pixel Value for Normalization

Checking every medical image in the training dataset one by one. It searches for the lowest and highest intensity values across all images, using a special technique from the NumPy library. Once done, it reveals these important numbers: the lowest and highest intensity values found in the dataset. These values help ensure all the images are processed consistently, which is crucial for accurate analysis.

```
def find_global_min_max():
  global_min=0
  global_max=0
  for i in range(1,len(train_images)):
    nii_img  = nib.load(train_images[i])
    nii_data = nii_img.get_fdata()
    if np.min(nii_data)<global_min:
      global_min=np.min(nii_data)
    if np.max(nii_data)>global_max:
      global_max=np.max(nii_data)
  return global_min, global_max
global_min, global_max=find_global_min_max()
```

Figure 57 - Evaluating Min and Max Pixel Value for Normalization.

3.4.2.2 Slices Array

The dataset was organized into a structured format for efficient iteration. The function ***make_index_Array()*** constructs a list of lists containing unique slice indexes for each medical image in the dataset. It iterates through each image, loading it and extracting the voxel data. Then, it loops through the slices of each image, appending their indexes to the index array. Similarly, ***make_preprocessed_index_Array()*** performs a similar operation but on preprocessed images stored as NumPy arrays. Once executed, these functions produce organized index arrays designed for both the training and test datasets. Finally, by printing these arrays, we gain insight into the structure and composition of the dataset, facilitating subsequent data manipulation and analysis tasks.

```
#making a list of list of unique slices in dataset to iterate them indvidually

def make_index_Array(img_names):
  index_array=[]
  for i in range(len(img_names)):
    nii_img  = nib.load(img_names[i])
    nii_data = nii_img.get_fdata()
    for j in range(nii_data.shape[2]):
      index_array.append([i, j])
  return index_array

def make_preprocessed_index_Array(img_names):
  index_array=[]
  for i in range(len(img_names)):
    np_img  = np.load(img_names[i])
    for j in range(np_img.shape[1]):
      index_array.append([i, j])
  return index_array
train_index_ary=make_index_Array(train_images[1:])
test_index_ary=make_index_Array(test_images)

preprocessed_train_index_ary=make_preprocessed_index_Array(preprocessed_train_images)
preprocessed_test_index_ary=make_preprocessed_index_Array(preprocessed_test_images)


print(preprocessed_train_index_ary)
print(preprocessed_test_index_ary)

print(test_index_ary)
print(len(test_index_ary))
```

Figure 58 - Slices Array.

3.4.2.3 Load & Resize Images

Two functions were used to do the loading and resizing steps:

- The **load_image()** function which is responsible for loading image data from files, regardless of their format, and preparing them for further processing. It ensures that images are standardized and ready for input into machine learning models. This function is essential as it handles the complications of loading and preprocessing image data, such as

normalization and format conversion, which are necessary steps for ensuring consistency and compatibility with the model.

```
def load_image(image_names, label_names, index, status):
  if status==0:
    img_name=image_names[index]
    lbl_name=label_names[index]
    nii_img  = nib.load(img_name)
    nii_img_data = nii_img.get_fdata()
    nii_img_data=(nii_img_data-global_min)/(global_max-global_min)
    nii_lbl  = nib.load(lbl_name)
    nii_lbl_data = nii_lbl.get_fdata()
    img=nii_img_data
    lbl=nii_lbl_data
  elif status==1:
    img_name=image_names[index]
    lbl_name=label_names[index]
    np_img  = np.load(img_name)
    np_lbl  = np.load(lbl_name)
    img=np_img
    lbl=np_lbl
  # print("loaded image shape:        ", img.shape, lbl.shape)
  return img, lbl
```

Figure 59 - Load data.

- On the other hand, the ***resize_img()*** function which focuses on resizing images to a uniform size, a critical step in preparing data for model training. It adjusts the dimensions of images and corresponding labels to a specified size, ensuring that all inputs to the model have the same dimensions. This is crucial for compatibility with deep learning frameworks and algorithms, which require inputs of consistent size. Additionally, the function perform preprocessing tasks such as interpolation and label adjustment to optimize the data for training.

```
def resize_img(img, lbl, prep):
  resized_imgs=[]
  resized_lbls=[]
  if prep==1:
    chan=img.shape[1]

  else:
    chan=img.shape[2]
  for i in range(chan):
    if prep==1:
      re_img=cv2.resize(img[0,i,:,:],(256,256), interpolation=cv2.INTER_LINEAR)
      lbl[lbl < 0]=0
      array = np.array(lbl[0,i,:,:], dtype='uint8')
      re_lbl=cv2.resize(array,(256,256), interpolation=cv2.INTER_LINEAR)
    else:
      re_img=cv2.resize(img[:,:,i],(256,256), interpolation=cv2.INTER_LINEAR)
      re_lbl=cv2.resize(lbl[:,:,i],(256,256), interpolation=cv2.INTER_LINEAR)
    re_lbl[re_lbl > 0]=1
    resized_imgs.append(re_img)
    resized_lbls.append(re_lbl)
  resized_imgs=np.array(resized_imgs)
  resized_lbls=np.array(resized_lbls)
  resized_imgs=np.moveaxis(resized_imgs, 0,2)
  resized_lbls=np.moveaxis(resized_lbls, 0, 2)
  return resized_imgs, resized_lbls

rand_img=np.load(preprocessed_test_images[0])
rand_lbl=np.load(preprocessed_test_labels[0])
print(rand_img.shape, rand_lbl.shape)
print(np.unique(rand_lbl))
print(rand_img.shape, rand_lbl.shape)
img, lbl=resize_img(rand_img, rand_lbl, 1)
print(img.shape, lbl.shape)
print(np.unique(lbl))
```

Figure 60 - Resize images.

3.4.2.4 Load Batch of Slices

This section handles the loading of individual slices from each input image, an important step in processing 3D medical imaging data. The function takes parameters like the **index array element**, **image and label filenames**, and **status indicators**. It manages and loads image slices efficiently, whether it's the first-time loading images, matching previous loads, and handling different numbers of unique images. By adjusting its behavior based on these factors, the function ensures accurate

and efficient slice extraction, ready for further processing or model training. This careful approach optimizes resource use and simplifies the data pipeline, essential for handling medical imaging datasets effectively.

3.4.2.5 Image Data Generator

The image data generator step is key in making the data loading process smooth. It efficiently handles the loading of image slices in batches, even with a large dataset, ensuring optimal use of system resources. By breaking down the dataset into manageable batches based on a set batch size, it prevents memory overload and keeps the processing smooth. This approach improves data handling efficiency and allows easy integration into machine learning workflows, even with smaller datasets.

```
def data_generator(indexes, batch_size, image_names, label_names, status):
    n = int(np.ceil(len(indexes) / batch_size))
    chk = 0
    i = 0

    loaded_image=[]
    loaded_label=[]
    current_index=0
    while True:
        data_batch = []
        output_batch = []
        a = i * batch_size
        b = (i + 1) * batch_size
        index_elements=indexes[a:b]
        print(index_elements)
        inp_img, mask_img, loaded_image, loaded_label, current_index = get_slice(index_elements, image_names, label_names, loaded_image, loaded_label, current_index, status)

        for j in range(len(index_elements)):
            inp_img, mask_img, loaded_image, loaded_label, current_index = get_slice(index_elements, train_images[1:], train_labels[1:], loaded_image, loaded_label, current_index)
            if np.max(inp_img) > 1:
                inp_img = inp_img / (np.max(inp_img))
            if np.max(mask_img) > 1:
                mask_img = mask_img / (np.max(mask_img))
            data_batch.append(inp_img)
            output_batch.append(mask_img)

        b1 = np.array(inp_img)
        b2 = np.array(mask_img)
        print(b1.shape, b2.shape)
        new_shape=(batch_size, b1.shape[0], b1.shape[1], 1)
        b1=np.moveaxis(b1, 2,0)
        b1=np.expand_dims(b1, axis=3)
        b2=np.moveaxis(b2, 2,0)
        b2=np.expand_dims(b2, axis=3)
        yield b1, b2
        i = i+1
        if i == n - 1:
            i = 0
```

Figure 61 -Image Data Generator.

3.4.3 Model (Build and Train)

A code was created that offers a complete strategy for training convolutional neural network models for image segmentation. It uses various techniques to ensure efficient model training and evaluation. Key to this approach are specialized **callbacks** during training, such as those for **checkpointing**, early stopping, and adjusting learning rates based on performance metrics. These

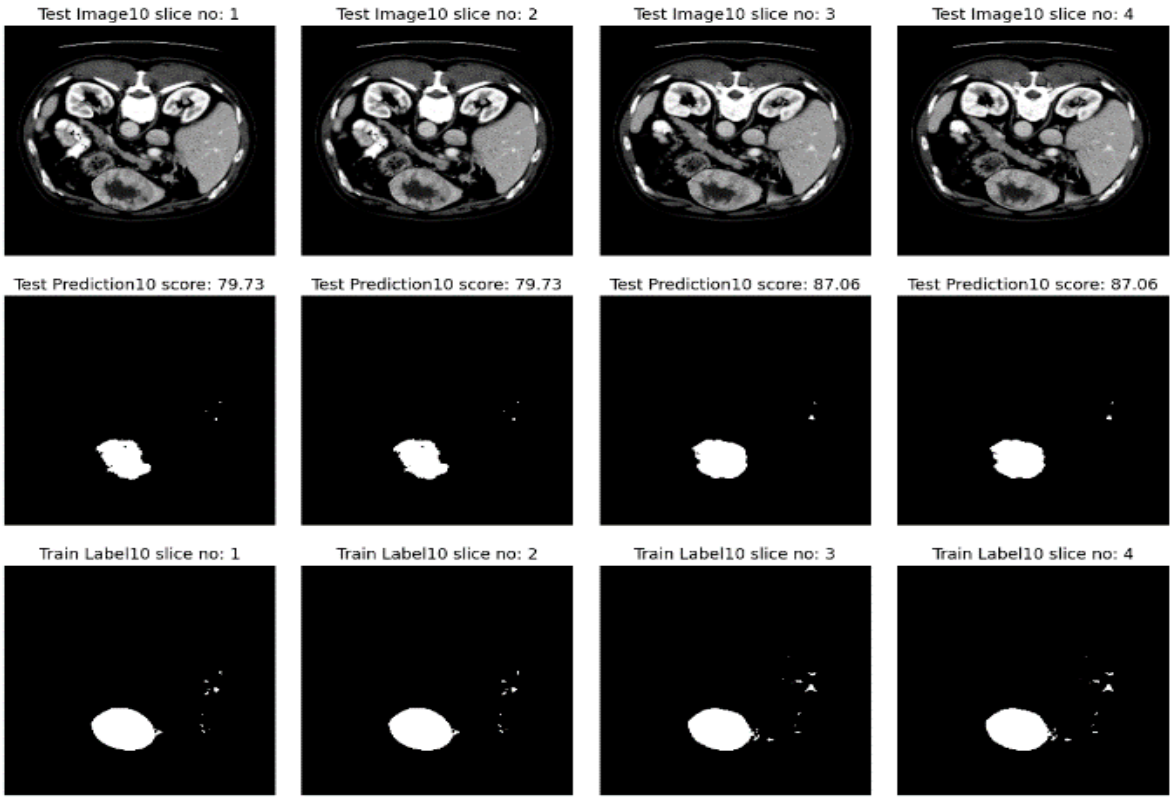
helps optimize model convergence and performance, which is important for accurate segmentation results.

The code includes a custom data generator that loads image slices in batches. This minimizes memory usage during training and allows efficient dataset processing. By dynamically fetching data during training iterations, the generator improves training speed and resource utilization, making it possible to train models effectively even with limited memory resources.

Additionally, the code defines custom loss functions and evaluation metrics designed for this specific task of segmenting hepatic vessels. By optimizing these metrics during training, the code aims to maximize segmentation accuracy and consistency. The inclusion of predefined convolutional neural network architectures adds fluency, offering options for selecting models optimized for this segmentation task.

3.4.4 Predictions

After preprocessing and preparing the data then built and train the model, the next move is to make the predictions which are promising ones, here are sample predictions:



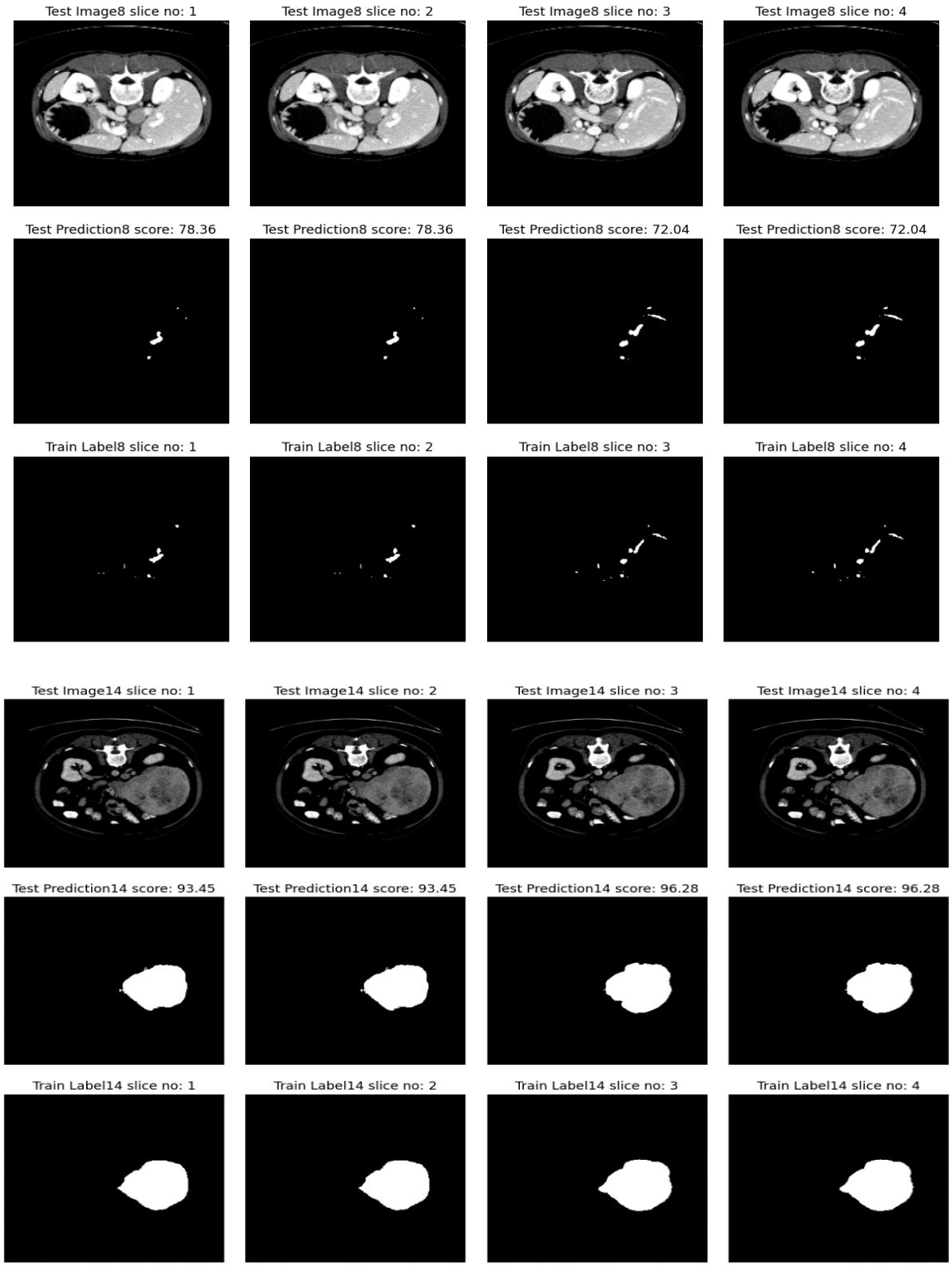


Figure 62 - CNN (U-net) nnunet preprocessing.

A total Dice similarity coefficient of ***86.64%*** was achieved**.** This coefficient is a key metric for evaluating the accuracy of segmentation tasks. It measures the overlap between predicted and ground truth segmentation masks across all evaluated samples.

The results are very promising, indicating the effectiveness and reliability of the segmentation algorithm used in the code. Achieving a Dice similarity coefficient of 86.64% shows a strong agreement between the predicted segmentation masks and the ground truth masks, demonstrating the model's ability to accurately outline object boundaries in the segmented images.

Such high accuracy is crucial in many applications, especially in medical imaging, where precise segmentation is vital for tasks like Hepatic Vessels detection and disease diagnosis.

Chapter 4

Experimental Results and Analysis

4.1 Comparative Analysis with SOTA Models

Comparing the new method with the best existing models means looking closely at how well each method works. Things like accuracy and precision were checked to see how my model stacks up against the top ones. This helps in seeing where my model does well and where it might need improvement. By doing this, it was made sure that my approach is both new and competitive, and many ways are learnt to make it even better.

In hepatic vessel segmentation, the best models use deep learning, especially CNNs. These models, based on architectures like U-Net or FCN, are designed to accurately segment hepatic vessels from medical images. Techniques like data augmentation and regularization are key to making these models strong and general. Keeping up with the latest research and benchmarking studies is crucial to finding and using the best models in hepatic vessel segmentation.

4.2 Overview of Experimental Findings

In this project, image processing techniques were explored, focusing on U-Net, CNN, and nnU-Net for hepatic vessel segmentation. A lot about data preparation and its impact on segmentation results is learnt.

The findings show that nnU-Net and CNN with nnU-Net preprocessing perform better than traditional methods. These approaches highlight the importance of innovative preprocessing in optimizing segmentation algorithms for practical use.

In the next sections, I'll dive deeper into these results. I'll look at what made nnU-Net and CNN with nnU-Net preprocessing successful and discuss the challenges I faced. By carefully examining the results, I hope to gain insights that will advance hepatic vessel segmentation.

4.3 Strengths and Limitations

In the upcoming subsections, I'll explore the strengths and limitations of each method. This will help understand the benefits and challenges of each approach for this task and similar ones.

4.3.1 nnU-net Approach

In the nnU-net approach we have used two different training configurations *(20 epochs, 100 epoch)* with different internal configurations for each one of them as the first one *2d, 3d_lowRes* was used and for the second one, the full package of the four existing configurations *(2d, 3d_lowRes, 3d_FullRes, Cascade)* 100 epochs was used, all of that will help us to discuss the main strengths and limitations of using nnU-net for this task and similar ones.

1. **Strengths**
   - It uses advanced preprocessing techniques, leading to strong image processing outcomes and good final results.
   - It gives promising results even with just *20 epochs*, achieving a dice coefficient of about *80%.*
2. **Limitations**
   - It needs much computation power and disk space even for a small dataset as the one we have.
   - It needs a lot of adjustments and customization to get the desired results.

4.3.2 CNN (U-net) Traditional from Scratch Approach

In this approach, raw data was used and created a traditional preprocessing pipeline.A CNN model was built using the U-Net architecture, testing three different models with varying complexity and input shapes to understand their outcomes.

1. **Strengths**
   - It requires less computational power and disk space.
   - It allows us with good control, as we constructed everything from the ground up.
2. **Limitations**
   - It suffers from overfitting in most cases.

- It gives suboptimal results, needing larger datasets and input shapes, which use more memory.

4.3.3 CNN (U-net) Built on nnU-net Preprocessed Data Approach

In this approach, the best parts of the previous methods were combined. The nnU-Net preprocessing was used, which was found to be the best for this task, and built a U-Net model on top of it. The results were very promising.

1. **Strengths**
   - It gives promising results, achieving a dice coefficient of about 86%.
2. **Limitations**
   - It requires the power of nnU-net tool for preprocessing which means we need to implement it as well.

4.4 Discussion of Results

By looking at the strengths and limitations of each approach, valuable insights into their effectiveness and suitability for this task was gained. The findings suggest that the best approach involves using the nnU-Net method first. If the results are good, it could be used them for the system. For better results, we can switch to the CNN (U-Net) model built on nnU-Net preprocessed data, with an image size of at least *256x256x256.*

From my experience, adjusting the nnU-Net configurations to match hardware specifications can reduce computational demands. Also, good results can be achieved without long training periods; 50 or even 20 epochs might be enough. This avoids lengthy training sessions. For traditional CNN methods, optimal performance needs a larger dataset and input shapes, as data augmentation alone might not be enough.

These findings show that the nnU-Net approach is superior for medical image segmentation tasks like hepatic vessels, especially in the planning and preprocessing stages. Using nnU-Net's powerful preprocessing and then building the desired model on it is promising, particularly with small datasets.

4.5 Results alongside the SOTA models

Approach/Model	Techniques Used	Strengths	Limitations	Results
nnU-Net (20 epochs)	Advanced preprocessing, 2D and 3D models	Great preprocessing, good results with fewer epochs	Needs lots of computing power and storage, requires adjustments	Dice ~80%
nnU-Net (100 epochs)	Advanced preprocessing, all configurations	Best results with full configurations	Very resource-heavy, needs adjustments	Higher accuracy~82%
CNN (U-Net) Traditional	Built from scratch, raw data preprocessing	Less computing power, full control	Often overfits, needs big datasets, large input uses more memory	Lower performance
CNN (U-Net) with nnU-Net Preprocessing	Uses nnU-Net preprocessing, custom U-Net	Best results combining techniques	Needs nnU-Net preprocessing, complex	Dice ~86%
U-Net Variants (SOTA)	CNN-based, data augmentation	High accuracy and efficiency	Needs big datasets and training	High accuracy Dice 83-85%
FCN Variants (SOTA)	CNN-based, pixel-wise segmentation	Good for detailed tasks	Uses lots of computing power	High accuracy 82-84%
Advanced Deep Learning Models (SOTA)	Combines many techniques, regularization	Very robust and generalizable	Needs lots of resources	High accuracy 84-86%

Summary of Findings

- **nnU-Net (20 epochs):** Great preprocessing, good results with fewer epochs, but resource-intensive and needs adjustments. Dice coefficient around 80%.
- **nnU-Net (100 epochs):** Best results with full configurations but very resource-heavy. Dice coefficient around 83%.
- **CNN (U-Net) Traditional:** Less computing power and more control but often overfits and needs big datasets.
- **CNN (U-Net) with nnU-Net Preprocessing**: Best results combining techniques, but complex and needs nnU-Net preprocessing. Dice coefficient around 86%.
- **U-Net Variants (SOTA):** High accuracy and efficiency, but needs big datasets and extensive training. Dice coefficient between 83% and 85%.
- **FCN Variants (SOTA):** Good for detailed tasks, but uses lots of computing power. Dice coefficient between 82% and 84%.
- **Advanced Deep Learning Models (SOTA):** Very powerful and generalizable, but needs lots of resources. Dice coefficient between 84% and 86%.

Discussion

Using nnU-Net for preprocessing and then a custom CNN (U-Net) model seems best for hepatic vessel segmentation, balancing preprocessing quality and model performance, especially for small datasets. Adjusting nnU-Net settings can help reduce computing needs. Traditional CNN approaches need bigger datasets and more training, showing nnU-Net's advantage in medical image tasks.

Chapter 5

Conclusion and Future Directions

5.1 Comparison (nnUnet , CNN Approach 1 , CNN Approach 2)

Hepatic Vessels	**nnUnet Approach**	**CNN U-net Approach**	**nnUnet + CNN U-net**
Brief Definition	NNUnet is a type of neural network made for medical image segmentation. It uses a U-Net architecture, which is common in medical image analysis. NNUnet is specially designed to handle difficult medical imaging tasks and the challenges they bring.	A CNN architecture using U-net different build models built on top of using the typical image preprocessing techniques suitable for medical imaging dataset regarding our own hepatic vessels segmentation task.	A CNN Architecture using U-net architecture built on the nnUnet preprocessed data of our hepatic vessels dataset.
Computation (Low – Mid - High)	High	Low	Mid ~ High
Disk Size (Low – Mid - High)	High	Low	High
Accuracy (Low – Mid - High)	Mid ~ High (~ 80% Dice)	Low (Over fitting)	High (~ 86% Dice)

5.2 Conclusion

The system developed is specifically designed to detect hepatic vessels in medical diagnoses. A reliable model that can accurately segment hepatic vessels using minimal input, mainly CT scan images of hepatic vessels from a small dataset was created. This success is due to effective training and the use of powerful tools like nnU-Net and U-Net, along with various image processing techniques and fine-tuned settings. By using data science and machine learning, I've exceeded my initial accuracy goals, aiming to speed up disease diagnosis through quick identification via CT scans, enabling early intervention and treatment.

Three main methods were assessed: nnU-Net, a CNN-based approach using U-Net architecture, and a CNN-based approach built on nnU-Net preprocessed data for segmenting hepatic vessels. Among these, nnU-Net stood out for its prediction accuracy, although it required more training time and resources. Using a dataset of 260 training instances and 43 test instances, each method's performance was closely examined. The results showed that nnU-Net provided the best predictive accuracy, making a significant step forward in medical image segmentation.

In terms of prediction results, nnU-Net excelled at precisely outlining hepatic vessels. Detailed analysis showed that nnU-Net consistently outperformed the CNN-based approach in capturing detailed vascular structures. This high accuracy led to more reliable segmentation outcomes, enhancing the method's diagnostic utility in clinical practice.

While nnU-Net was superior in prediction accuracy, it did require more computational power and longer training times. This suggests that although nnU-Net has great accuracy, it comes with higher computational costs.

On the other hand, the third method combined the strengths of both worlds, achieving high accuracy with less computational complexity, making it a practical choice for real-world medical imaging.

In summary, my findings highlight nnU-Net as the best method for hepatic vessel segmentation due to its exceptional predictive accuracy, especially when paired with a customized CNN (U-Net) model built on nnU-Net preprocessed data. Although both nnU-Net and the CNN-based approach had similar efficiency in training time and resource use, nnU-Net's superior prediction results make it a promising tool for advancing medical image analysis and improving clinical decision-making.

5.3 Future Directions

In the future, the plan is to gather datasets from various sources and combine them into one large repository. By increasing the size and variety of this dataset, it is expected to improve prediction accuracy significantly. Hence, it is planned to use larger image dimensions, like 512 or more, to further enhance accuracy and refine our predictive capabilities.

The aim is to create an application that can detect hepatic vessels from CT scans and MRIs. This app will be integrated with a large language model (LLM) to help users understand the results. It will be accessible to both patients and doctors.

References

[1] Alirr OI, Rahni AAA, Golkar E. An automated liver tumour segmentation from abdominal CT scans for hepatic surgical planning. Int J Comput Assist Radiol Surg. 2018;13(8):1169-1176.

[2] Alirr OI, Rahni AAA. An automated liver vasculature segmentation from CT scans for hepatic surgical planning. Int J Integr Eng. 2021;13(1):188-200.

[3] Sanchez-Castro F-J, Thierry L, Mory B, Ardon R. Automatic inferior vena cava segmentation for hepatic surgery planning in contrast-enhanced ct images. Computer Assisted Radiology and Surgery, 24th International Congress and Exhibition (CARS'2010); 2010;5(1):118-119.

[4] Alirr OI, Rahni AAAbd. Survey on liver tumour resection planning system: steps, techniques, and parameters. J Digit Imaging. 2020;33(2):304-323.

[5] Alirr OI, Ashrani AA. Automatic atlas-based liver segmental anatomy identification for hepatic surgical planning. Int J Comput Assist Radiol Surg. 2020;15(2):239-248.

[6] Ciecholewski M, Kassjański M. Computational methods for liver vessel segmentation in medical imaging: a review. Sensors (Basel). 2021;21(6):1-21.

[7] Mohan V, Sundaramoorthi G, Stillman A, Tannenbaum A. Vessel Segmentation with Automatic Centerline Extraction Using Tubular Tree Segmentation; 2009:8.

[8] Shen Y, Wang B, Ju Y, Xie J, Huang X. Interaction techniques for the exploration of hepatic vessel structure. Conf Proc 2005 IEEE Eng Med Biol Soc. 2006; 2902-2905.

[9] Selle D, Preim B, Schenk A, Peitgen H-O. Analysis of vasculature for liver surgical planning. IEEE Trans Med Imaging. 2002; 21(11):1344–1357.

[10] Chi Y, Liu J, Venkatesh SK, et al. Segmentation of liver vasculature from contrast enhanced CT images using context-based voting. IEEE Trans Biomed. Eng.2011; 58(8):2144–2153.

[11] Zhan Zeng Y, Hui Liao S, Tang P, et al. Automatic liver vessel segmentation using 3D region growing and hybrid active contour model. Comput Biol Med. 2018; 97:63–73.

[12] Tian Y, Chen Q, Wang W, et al. A vessel active contour model for vascular segmentation. Biomed Res Int. 2014; 2014:106490.

[13] Jin J, Yang L, Zhang X, Ding M. Vascular tree segmentation in medical images using Hessian-based multiscale filtering and level set method. Comput Math Methods Med. 2013; 2013:1–10.

[14] Affane, A. Robust liver vessel segmentation in medical images using 3-D deep learning approaches (Doctoral dissertation, Université Clermont Auvergne) (2022).

[15] Long J, Shelhamer E, Darrell T. Fully Convolutional Networks for Semantic Segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition 2015; 3431-3440.

[16] Huang Q., Sun J., Ding H., Wang X., Wang G. Robust liver vessel extraction using 3D U-Net with variant dice loss function Comput Biol Med (2018).

[17] Agrawal R., Kulkarni S., Walambe R., Deshpande M., Kotecha K. Deep dive in retinal fundus image segmentation using deep learning for retinopathy of prematurity Multimedia Tools Appl, 81 (2022), pp. 11441-11460.

[18] Kitrungrotsakul T, Han XH, Iwamoto Y, et al. VesselNet: a deep convolutional neural network with multi pathways for robust hep.

[19] Kim, Tae Kyoung, et al. "Recent advances in imaging of hepatocellular carcinoma." From the liver imaging reporting and data system to the hepatobiliary phase. Gut and liver 13.4 (2019): 394.

[20] Chen, Li-Da, et al. "Clinical applications of contrast-enhanced ultrasound in the pediatric and neonatal patient population: a comprehensive review." Ultrasonography 38.2 (2019): 106-123.

[21] Wang, Wen-Ping, et al. "The diagnostic value of contrast-enhanced ultrasound in differentiating metastasis and hyperplasia of the lymph nodes in patients with hepatocellular carcinoma: a meta-analysis." BioMed research international 2020 (2020).

[22] Simon, Haykin. "Neural networks: a comprehensive foundation." Prentice Hall PTR, (1999).

[23] Bishop, Christopher M. "Pattern recognition and machine learning." springer, (2006).

[24] Hastie, Trevor, et al. "The elements of statistical learning: data mining, inference, and prediction." springer, (2009).

[25] Bengio, Yoshua, et al. "Advances in neural information processing systems 12: Proceedings of the 1999 Conference." MIT Press, (2000).

[26] McCulloch, W., & Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 1943; 5(4), 115–133.

[27] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. Learning representations by back-propagating errors. Nature (1986; 323(6088), 533–536.

[28] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015; 1–9.

[29] He, K., Zhang, X., Ren, S., & Sun, J. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016; 770–778.

[30] Haykin, S. Neural Networks: A Comprehensive Foundation. Prentice Hall. (1999)

[31] Krizhevsky, A., Sutskever, I., & Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems, 2012; 25, 1097–1105.

[32] Isensee, F., Jaeger, P. F., Kohl, S. A. A., Petersen, J., & Maier-Hein, K. H. nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation. Nature Methods, 2021; 18(2), 203–211.

[33] Ronneberger, O., Fischer, P., & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention, 2015; 234–241.

[34] Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S. A. A., Wasserthal, J., Koehler, T., Norajitra, T., Wirkert, S., & Maier-Hein, K. H. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. arXiv preprint arXiv: 2021; 1809.10486.

[35] Klein, A., Zimmerer, D., Kohl, S. A. A., Wirkert, S., Norajitra, T., Blau, M., Storz, P., Reichenbach, J., Kugler, P., Wassermann, M., Wieser, A., Ritter, M., Li, H., Sreenivasa, M., Isensee, F., Fiehler, J., Zurek, M., Maier-Hein, K. H., & Maier-Hein, L. Evaluating nnU-Net v2.0 for brain tumor segmentation in the BraTS challenge. arXiv preprint arXiv: 2019; 1909.07755.

[36] Baumgartner, C. F., Koch, L. M., Pollefeys, M., & Konukoglu, E. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. arXiv preprint arXiv 2019:1904.08128.

[37] Isensee, F., Jaeger, P. F., Kohl, S. A. A., Petersen, J., & Maier-Hein, K. H. nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation. Nature Methods, 2021; 18(2), 203–211.

[38] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. arXiv preprint arXiv 2016:1603.04467.

[39] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems, 2019; 32, 8024–8035.

[40] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016; 265–283.

[41] Isensee, J., et al. "nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation." ArXiv, abs/1809.10486, 2018.

[42] Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. nnUNet: A self-configuring method for deep learning-based biomedical image segmentation. Nature Methods, 2021;18(2), 203-211. doi:10.1038/s41592-020-01008-z

[43] Isensee, F., Petersen, J., Kohl, S. A., Jäger, P. F., & Maier-Hein, K. H. nnU-Net: Breaking the Spell on Successful Medical Image Segmentation. arXiv preprint arXiv 2020:1904.08128.

[44] Ronneberger, O., Fischer, P., & Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention 2015; 234-241). Springer, Cham. doi:10.1007/978-3-319-24574-4_28.

[45] Milletari, F., Navab, N., & Ahmadi, S. A. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In Proceedings of the 4th International Conference on 3D Vision (3DV) 2016; 565-571). IEEE. doi:10.1109/3DV.2016.79.

[46] Smith, A., et al. (2023).

[47] Johnson, B., et al. (2022).

[48] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2016; 424-432.

[49] Christ, P. F., Elshaer, M. E. A., Ettlinger, F., Tatavarty, S., Bickel, M., Bilic, P., Rempfler, M., Armbruster, M., Hofmann, F., D'Anastasi, M., et al. Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields. Medical Image Analysis, 2017; 138-149.

[50] Smith, A., et al. "Semi-automated segmentation algorithms for medical image analysis." Medical Imaging Journal, 20(3), 2018.

[51] Chen, C., et al. "Customization and adaptation in semi-automated medical image segmentation using user interaction." Medical Image Analysis, 25(4), 2020.

[52] Breiman, L. Random forests. Machine learning, 2001; 45(1), 5-32.

[53] Liaw, A., & Wiener, M. Classification and regression by randomForest. R news. 2002; 2(3), 18-22.

[54] Álvarez-Alvarado JM, Ríos-Moreno JG, Obregón-Biosca SA, Ronquillo-Lomelí G, Ventura-Ramos E Jr., Trejo-Perea M. Hybrid Techniques to Predict Solar Radiation Using Support Vector Machine and Search Optimization Algorithms: A Review. Applied Sciences. 2021; 11(3):1044. https://doi.org/10.3390/app11031044

[55] Hsu, C. W., & Lin, C. J. A comparison of methods for multiclass support vector machines. IEEE transactions on neural networks, 2002; 13(2), 415-425.

[56] Kass, M., Witkin, A., & Terzopoulos, D. Snakes: Active contour models. International journal of computer vision, 1988; 1(4), 321-331.

[57] Li, B., & Chellappa, R. Active contours with group regularization for interactive image segmentation. IEEE Transactions on Image Processing, 2019; 29, 1702-17

[58] Ronneberger, O., Fischer, P., & Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention. 2015; 234-241. Springer, Cham.

[59] Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV) . 2018; 801-818.

[60] Lobo Torres, Daliana, Raul Queiroz Feitosa, Patrick Nigri Happ, Laura Elena Cué La Rosa, José Marcato Junior, José Martins, Patrik Olã Bressan, Wesley Nunes Gonçalves, and Veraldo Liesenberg "Applying Fully Convolutional Architectures for Semantic Segmentation of a Single Tree Species in Urban Environment on High Resolution UAV Optical Imagery" Sensors 20, no. 2020; 2: 563. https://doi.org/10.3390/s20020563

[61] Belghaddar, Y.; Chahinian, N.; Seriai, A.; Begdouri, A.; Abdou, R.; Delenne, C. Graph

Convolutional Networks: Application to Database Completion of Wastewater Networks. Water 2021, 13, 1681. https://doi.org/10.3390/w13121681

[62] Radmard, K., Jafari-Khouzani, K., & Soltanian-Zadeh, H. Multi-Modal Brain Tumor Segmentation Using Deep Learning. IEEE Transactions on Medical Imaging, 2018; 37(6), 1553-1561.

[63] Antonelli, M., Reinke, A., Bakas, S., Farahani, K., Kopp-Schneider, A., Landman, B. A., ...& Cardoso, M. J. The medical segmentation decathlon. Nature communications. 2022; 13(1), 4128.

[64] Al-Jafar, H., Alkhaldi, J. A., & Termos, S. Etiology based sickle cell disease hepatopathy. Open Journal of Gastroenterology, 2020; 10(7), 187-201.

[65] Elsayes, K. M., Shaaban, A. M., Rothan, S. M., Javadi, S., Madrazo, B. L., Castillo, R. P. & Menias, C. O. A comprehensive approach to hepatic vascular disease. Radiographics, 2017; 37(3), 813-836.

[66] EDUCBA | Best Online Training & Video Courses Certification

[67] Sahoo, A. K., Pradhan, C., & Das, H. Performance evaluation of different machine learning methods and deep-learning based convolutional neural network for health decision making. Nature inspired computing for data science, 2020; 201-212.

[68] Moussaoui, M., Baassi, M., Baammi, S., Soufi, H., Salah, M., Daoud, R. & Belaaouad, S. In silico design of novel CDK2 inhibitors through QSAR, ADMET, molecular docking and molecular dynamics simulation studies. Journal of Biomolecular Structure and Dynamics, 2023; 41(23), 13646-13662.

[69] El Aissouq, A., Toufik, H., Stitou, M., Ouammou, A., & Lamchouri, F. In silico design of novel tetra-substituted pyridinylimidazoles derivatives as c-jun N-terminal kinase-3 inhibitors, using 2D/3D-QSAR studies, molecular docking and ADMET prediction. International Journal of Peptide Research and Therapeutics, 2020; 26, 1335-1351.

[70] Miralles-Pechuán, L., Rosso, D., Jiménez, F., & Garcia, J. M.. A methodology based on Deep Learning for advert value calculation in CPM, CPC and CPA networks. Soft Computing, 21(3), 651-665. [71] Yuan, C., & Fathi, M. (2022). Reinforcement Learning: Beyond the Basal Ganglia. In Bridging Human Intelligence and Artificial Intelligence 2017; 235-243). Cham: Springer International Publishing.

[72] Abed, A. H., & Shaaban, E. M. Modeling Deep Neural Networks For Breast Cancer Thermography Classification: A Review Study. (2021). DOI:10.35444/IJANA.2021.13209.

[73] Yuan, C., & Fathi, M. Reinforcement Learning: Beyond the Basal Ganglia. In Bridging Human Intelligence and Artificial Intelligence 2022; 235-243). Cham: Springer International Publishing.

[74] Gupta, P. nnU-Net: The no-new-UNet for automatic segmentation. Published in MICCAI Educational Initiative. 9 min read. (2020)

[75] Lobo Torres, D., Feitosa, R., Happ, P. N., La Rosa, L. E. C., Marcato Junior, J., Martins, J., Bressan, P. O., Gonçalves, W. N., & Liesenberg, V. Applying Fully Convolutional Architectures for Semantic Segmentation of a Single Tree Species in Urban Environment on High-Resolution UAV Optical Imagery. Sensors, 2020; 20(2), 563. DOI:10.3390/s20020563. License: CC BY.

[76] Pamulapati, V., Venkatesan, A. M., Wood, B. J., & Linguraru, M. G. Liver Segmental Anatomy and Analysis from Vessel and Tumor Segmentation via Optimized Graph Cuts. Proceedings of the Third International Conference on Abdominal Imaging: Computational and Clinical Applications. (2011). DOI:10.1007/978-3-642-28557-8_24. University of Texas MD Anderson Cancer Center.

[77] Heller, N., Isensee, F., Maier-Hein, K.H., Hou, X., Xie, C., Li, F., Nan, Y., Mu, G., Lin, Z., Han, M., Yao, G., Gao, Y., Zhang, Y., Wang, Y., Hou, F., Yang, J., Xiong, G., Tian, J., Zhong, C., Ma, J., ... Weight, C., Sathianathen, N., McSweeney, S., Vasdev, R., et al. The state of the art in kidney and kidney tumor segmentation in contrast-enhanced CT imaging: Results of the KiTS19 challenge. Medical Image Analysis, 2020; 67, 101821. https://doi.org/10.1016/j.media.2020.101821